HDF-EOS5 Data Model, File Format and Library

1. Status of this Memo

This is a description of a Recommended Standard to the Standards Process Group

2. Change Explanation

November, 2007 – changes made based on recommendations by the HDF Group.

3. Copyright Notice

4. Abstract

HDF-EOS is a software library designed to support NASA Earth Observing System (EOS) science data. HDF is the Hierarchical Data Format developed by the National Center for Supercomputing Applications. Specific data structures which are containers for science data are: Grid, Point, Zonal Average and Swath. These data structures are constructed from standard HDF data objects, using EOS conventions, through the use of a software library. A key feature of HDF-EOS is a standard prescription for associating geolocation data with science data through internal structural metadata. The relationship between geolocation and science data is transparent to the end-user. Instrument and data type-independent services, such as subsetting by geolocation, can be applied to files across a wide variety of data products through the same library interface. The library is extensible and new data structures can be added. This document describes a proposed standard for HDF-EOS5 Grid and Swath structures, which is based on the HDF5 data model and file format, provided by the HDF Group. The HDF Group was part of the National Center for Supercomputing Applications (NCSA) until July 2006, at which time it began full operations as a non-profit 501(c)(3) company.

Table of Contents

5. Introduction

The Hierarchical Data Format (HDF) was selected by NASA as the format of choice for standard science product archival and distribution for the Earth Observing System (EOS) Project. HDF is a file format and I/O library that was originally developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign to provide a portable storage mechanism for supercomputer simulation results. (HDF5 Users Guide, National Center for Supercomputing Applications, U. of Illinois, Urbana-Champaign, 2005)

HDF5 files consist of a directory and a collection of data objects. Every data object has a directory entry, containing a pointer to the data object location, and information defining the datatype (much more information about HDF5 can be found in the NCSA documentation (HDF5 API Specification Reference Manual,http://hdfgroup.org./HDF5/doc/RM_H5Front.html). Many of the NCSA defined datatypes map well to EOS datatypes. Examples include raster images, multi-dimensional arrays, and text blocks. There are other EOS datatypes, however, that do not map directly to NCSA datatypes, particularly in the case of geolocated datatypes. Examples include projected grids, satellite swaths, and field campaign or point data. Therefore, some additions to conventional HDF5 datatypes were required to fully support these datatypes.

To bridge the gap between the needs of EOS data products and the capabilities of HDF, new EOS specific datatypes – *Point*, *Swath*, and *Grid* – were defined within the HDF framework. Each of these new datatypes was constructed using conventions for combining standard HDF datatypes and is supported by an Application Programming Interface (API) which aids the data product user or producer in the application of the conventions. The APIs allow data products to be created and manipulated in ways appropriate to each datatype, without regard to or the users needing to manipulate the underlying HDF objects.

The sum of these APIs comprise the HDF-EOS library. The *Point* interface is designed to support data that has associated geolocation information, but is not organized in any well defined spatial or temporal way. The *Swath* interface is tailored to support time-ordered data such as satellite swaths (which consist of a time-ordered series of scanlines), or profilers (which consist of a time-ordered series of profiles). The *Grid* interface is designed to support data that has been stored in a rectilinear array based on a well defined and explicitly supported projection. Profile data is Swath-like data without geo-referencing information attached.

The original HDF-EOS library was constructed beginning in 1995, using the version of HDF available at the time, HDF4. The HDF-EOS version was called HDF-EOS2, the version number being a historical artifact. In 2001, a completely new version of HDF was introduced, HDF5. This library was based on a different data model (HDF5 for HDF4 Users: a short guide, National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, December 3, 2002, http://www.hdfgroup.uiuc.edu/papers/papers/h4toh5/HDF5forHDF4Users.pdf) and had an interface which was very different than that of HDF4. HDF-EOS was upgraded to support HDF5 and is called HDF-EOS5. This new version of HDF-EOS supports the same data model as does HDF-EOS2 and maintains the HDF-EOS2 interface to the maximum extent possible. Besides the three data types mentioned above, i.e. Grid, Swath, and Point, HDF-EOS5 also supports “Zonal Average” data type which is basically a swath like datatype without geolocation mapping.

At the present time, most EOS data products, several petabytes worth (10^{15}), are produced and stored in HDF-EOS2. A growing volume of data is being created in HDF-EOS5 and both libraries are supported by NASA. Production of EOS data will continue so long as instruments continue to operate.

This document presents a proposed standard for HDF-EOS5 Grid and Swath structures. Point and Zonal Average (ZA) structures will not be addressed.

5.1 What is HDF-EOS5?

HDF-EOS5 has three components: (1) a data model which describes Grid, Point, Swath , and ZA structures, (2) a file format and (3) an Application Programming Interface (API) which implements the data model and enforces the standard.

The *data model* provides the format to allow creation, storage, and access to Grid, Point, Swath, and ZA structures. It specifies the packaging of geolocation data, science data, and metadata. The data model for Grid and Swath data is described in Section 6.

The *file format* describes how the HDF-EOS5 data structures are represented in basic HDF5 objects. These objects in turn specify how the structures are stored in memory, or on disk or other media. HDFEOS5 is self-describing in that the internal structure of the files is described within the file. The file format, which is represented by the HDF5 file format is described in Section 7.

The API implements the data model in a number of programming languages, including C, FORTRAN and C++. This library, which is represented by an Application Programming Interface (API) is described in Section 8.

5.2 Motivation for Proposing Standardization

HDF5 is the underlying format for HDF-EOS5. HDF-EOS is the standard format and I/O library for the Earth Observing System (EOS) Data and Information System (EOSDIS). EOSDIS is the data system supporting a coordinated series of polar-orbiting and low inclination satellites for long-term global observations of the land surface, biosphere, solid Earth, atmosphere, and oceans. HDF-EOS2 is the standard for the EOS Terra and Aqua missions and HDF-EOS5 is the standard for the EOS Aura mission. There is a possibility that HDF-EOS2 files will be converted to HDF-EOS5 during future re-processing.

We note several successes using the HDF-EOS standard. The EOS MODIS instrument team used Swath and Grid formats for its' science product storage and distribution format. Science products comprised many disciplines, including Oceanographic, Land and Atmospheric data. The team had more than thirty Principle Investigators supplying data processing algorithms and code. A single integrator at NASA Goddard Space Flight Center was charged with implementing the algorithms, integrating processing code and formatting output data. Use of HDF-EOS as a team saved considerable code development and schedule.

A second example of efficiency associated with use of the HDF-EOS standard was found in the work of the EOS Aura team. A standard was developed and adopted for all four Aura instruments. Data produced in HDF-EOS5, were than in common format across science data produced by platform instruments.

The EOS Atmospheric Infrared Sounder (AIRS) instrument is a facility instrument with dozens of NASA and NOAA users. This is a profiling instrument, which stores data in a very different format than does MODIS, which is an imaging instrument. The team comprised of many Principle Investigators, each generating their own production algorithms and data products, successfully packaged its' products in the HDF-EOS format.

The next major Earth observing system, NPOESS will use HDF5 to store and distribute its data. There will be considerable overlap in the kinds of measurements made by EOS and by NPOESS instruments. There will be a need to compare data to develop a consistent long term data record. Community standardization of both HDF5 and HDF-EOS5 extensions will be of great importance. (HDF5 Draft Community Standard, ESE RFC, 2005)

EOS data stored in HDF-EOS2 and HDF-EOS5 are of fundamental importance to current and future research on global climate change and other physical, chemical and biological processes impacting our earth's environment. ESE standardization of HDFEOS5 will help to accelerate its adoption among the earth science communities, and many others as well, both through an increase in the number of developers writing to the specification and using the API, and through an increase in the number of those providing their data in HDFEOS5. We don't propose an ESE standard for HDF-EOS2, but refer the reader to numerous documents describing the format. (HDF-EOS5 Interface Based on HDF5, 2005) We again note that the HDF-EOS2 data model for Grid and Swath data is the same as that of HDF-EOS5. The API of HDF-EOS 5 has the same look and feel of its predecessor, but carry parameter additions necessitated by major differences between HDF4 and HDF5.

ESE standardization will also validate HDF5 to vendors of software applications important to users of HDF-EOS5, increasing the likelihood that these vendors will support the standard.

6. HDF-EOS5 Data Model

6.1 SWATH Data Model

The Swath concept for HDF-EOS is based on a typical satellite swath, where an instrument takes a series of scans perpendicular to the ground track of the satellite as it moves along that ground track. Figure 6.1-1 below shows this traditional view of a swath.

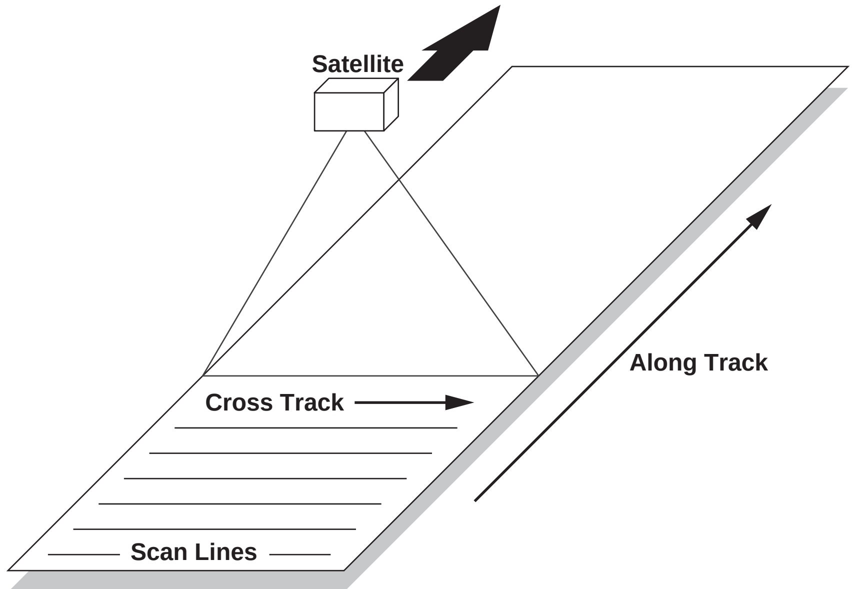

Figure 6.1-1. A Typical Satellite Swath: Scanning Instrument

Another type of data that the Swath is equally well suited to arise from a sensor that measures a vertical profile, instead of scanning across the ground track. The resulting data resembles a standard Swath tipped up on its edge. Figure 6.1-2 shows how such a Swath might look.

In fact, the two approaches shown in Figures 6.1-1 and 6.1-2 can be combined to manage a profiling instrument that scans across the ground track. The result would be a three dimensional array of measurements where two of the dimensions correspond to the standard scanning dimensions (along the ground track and across the ground track), and the third dimension represents a height above the Earth or a range from the sensor. The "horizontal" dimensions can be handled as normal geographic dimensions, while the third dimension can be handled as a special "vertical" dimension.

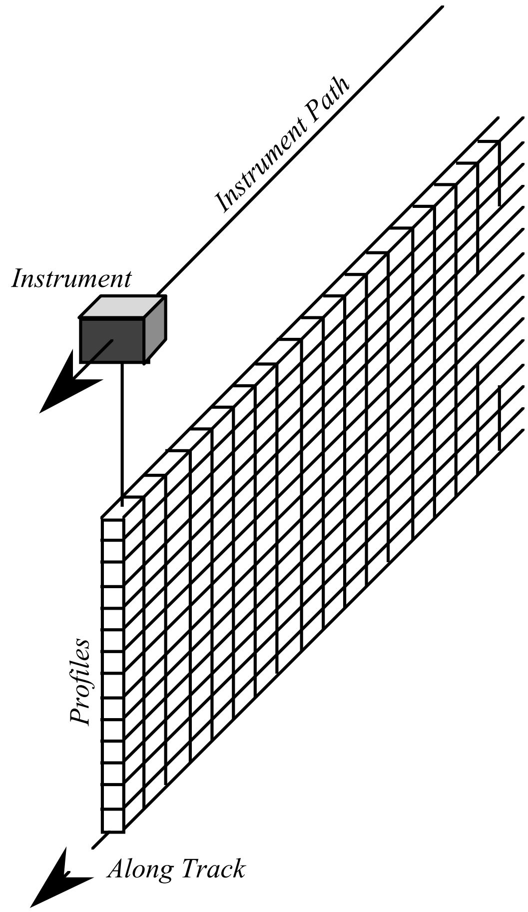

Figure 6.1-2. A Swath Derived from a Profiling Instrument

A standard Swath is made up of four primary parts: data fields, geolocation fields, dimensions, and dimension maps. An optional fifth part called an index can be added to support certain kinds of access to Swath data. Each of the parts of a Swath is described in detail in the following subsections.

6.1.1 Data Fields

Data fields are the main part of a Swath from a science perspective. Data fields usually contain the raw data (often as *counts*) taken by the sensor or parameters derived from that data on a value-for-value basis. All the other parts of the Swath exist to provide information about the data fields or to support particular types of access to them. Data fields typically are two-dimensional arrays, but can have as few as one dimension or as many as eight, in the current library implementation. They can have valid 32 and 64-bit floating point numbers, 8,16,32 and 64-bit integers, etc.

6.1.2 Geolocation Fields

Geolocation fields allow the Swath to be accurately tied to particular points on the Earth's surface. To do this, the Swath interface requires the presence of at least a time field ("Time") or a latitude/longitude field pair (“Latitude”[1] and “Longitude”). Geolocation fields must be either one- or two-dimensional and can have 32 or 64-bit types. The “Time” field is always in TAI format.(International Atomic Time, see SDP Toolkit Users Guide for the ECS Project)

Figure 6.1-3 shows a ‘data view’ of a swath structure. Here, the track parameter can be represented by time.

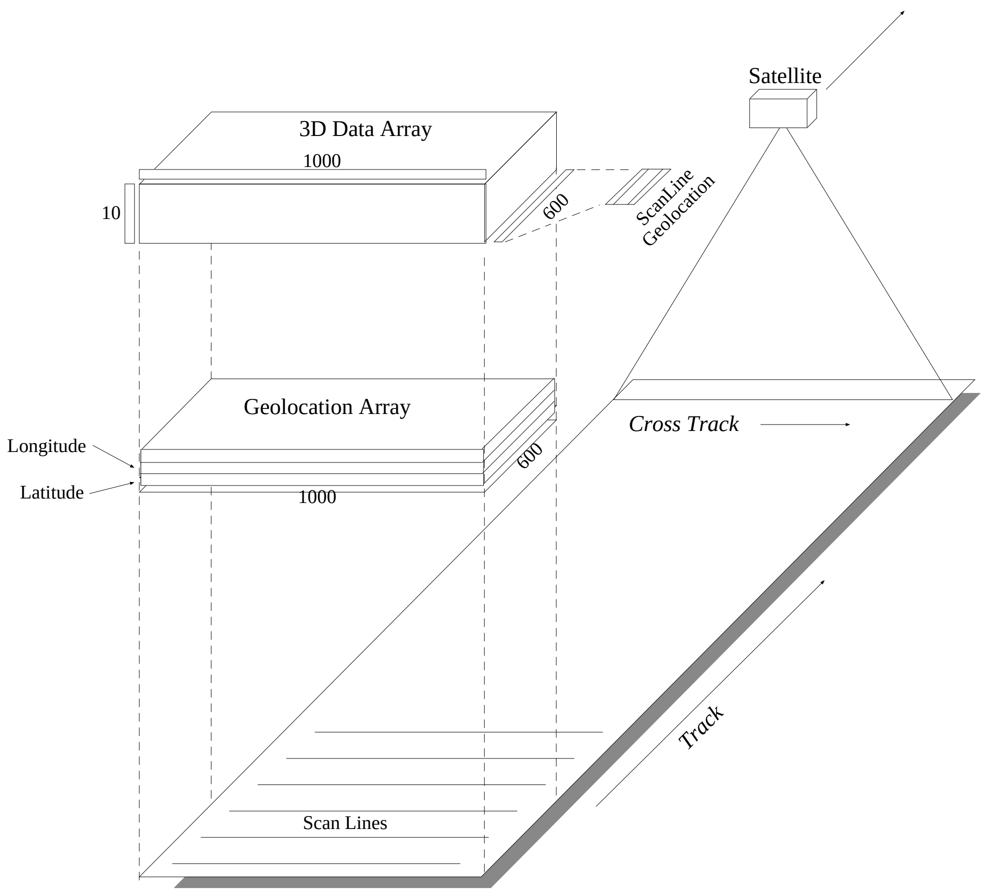

Figure 6.1-3 Conceptual View of Example Swath, with 3D Time/Geolocation Array, and Geolocation Table.

[1] “Colatitude” may be substituted for “Latitude.”

6.1.3 Dimensions

Dimensions define the axes of the data and geolocation fields by giving them names and sizes. In using the library, dimensions must be defined before they can be used to describe data or geolocation fields. The defined dimensions are stored in the structure metadata.

Every axis of every data or geolocation field, then, must have a dimension associated with it. However, there is no requirement that they all be unique. In other words, different data and geolocation fields may share the same named dimension. In fact, sharing dimension names allows the Swath interface to make some assumptions about the data and geolocation fields involved which can reduce the complexity of the file and simplify the program creating or reading the file.

6.1.4 Dimension Maps

Dimension maps are the glue that holds the Swath together. They define the relationship between data fields and geolocation fields by defining, one-by-one, the relationship of each dimension of each geolocation field with the corresponding dimension in each data field. In cases where a data field and a geolocation field share a named dimension, no explicit dimension map is needed. In cases where a data field has more dimensions than the geolocation fields, the "extra" dimensions are left unmapped. Like the dimensions the dimension maps are stored in the structure metadata.

In many cases, the size of a geolocation dimension will be different from the size of the corresponding data dimension. To take care of such occurrences, there are two pieces of information that must be supplied when defining a dimension map: the *offset* and the *increment*. The offset tells how far along a data dimension that must be traversed to find the first point to have a corresponding entry along the geolocation dimension. The increment tells how many points to travel along the data dimension before the next point is found for which there is a corresponding entry along the geolocation dimension. Figure 6.1-4 depicts a normal dimension map.

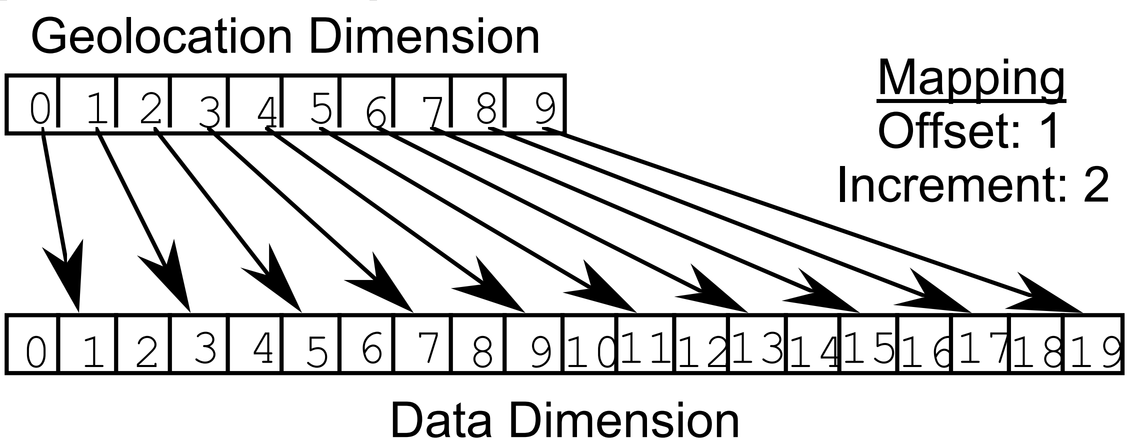

Figure 6.1-4. A "Normal" Dimension Map

The “data skipping” method described above works quite well if there are fewer regularly spaced geolocation points than data points along a particular pair of mapped dimensions of a Swath. It is conceivable, however, that the reverse is true – that there are more regularly spaced geolocation points than data points. In that event, both the offset and increment should be expressed as negative values to indicate the reversed relationship. The result is shown in Figure 6.1-5. Note that in the reversed relationship, the offset and increment are applied to the geolocation dimension rather than the data dimension.

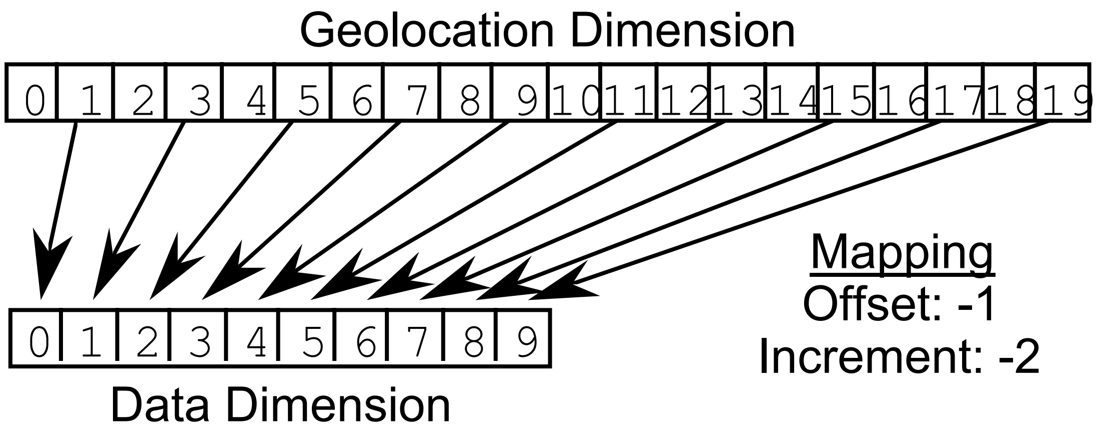

Figure 6.1-5. A “Backwards” Dimension Map

6.1.5 HDF5 Objects in HDF-EOS 5 Swath Objects

Figure 6.1-6 shows the relationship between HDF-EOS5 and HDF5 objects. The “SWATHS” object is an HDF5 group object (see Figure 7.2-1) that contains one or more Swath groups with user defined names and optional HDF5’s supporting attribute objects. The Swath groups in turn contain three HDF5 groups named “Data Fields”, “Geolocation Fields”, and “Profile Fields”. These three groups, like the “SWATHS” group, have reserved names and are internally created by HDF-EOS. Besides the group attributes, which again are simply HDF5 attribute objects, these groups hold HDF5 datasets containing science data, and geolocation field data for the “Latitude”, “Longitude”, or “Time”. Each data field or geolocation field also may contain optional dataset related attributes, called local attributes, such as fillvalue, units, etc. Please note that HDF-EOS5 always creates the attribute “_FillValue” for every data field and sets its value to zero. The default zero value is replaced with user provided value when user calls appropriate fillvalue setting routine.

The Profile fields shown in this figure are profile swath fields that are described in section 2.1 and are depicted in Figure 6.1-2.

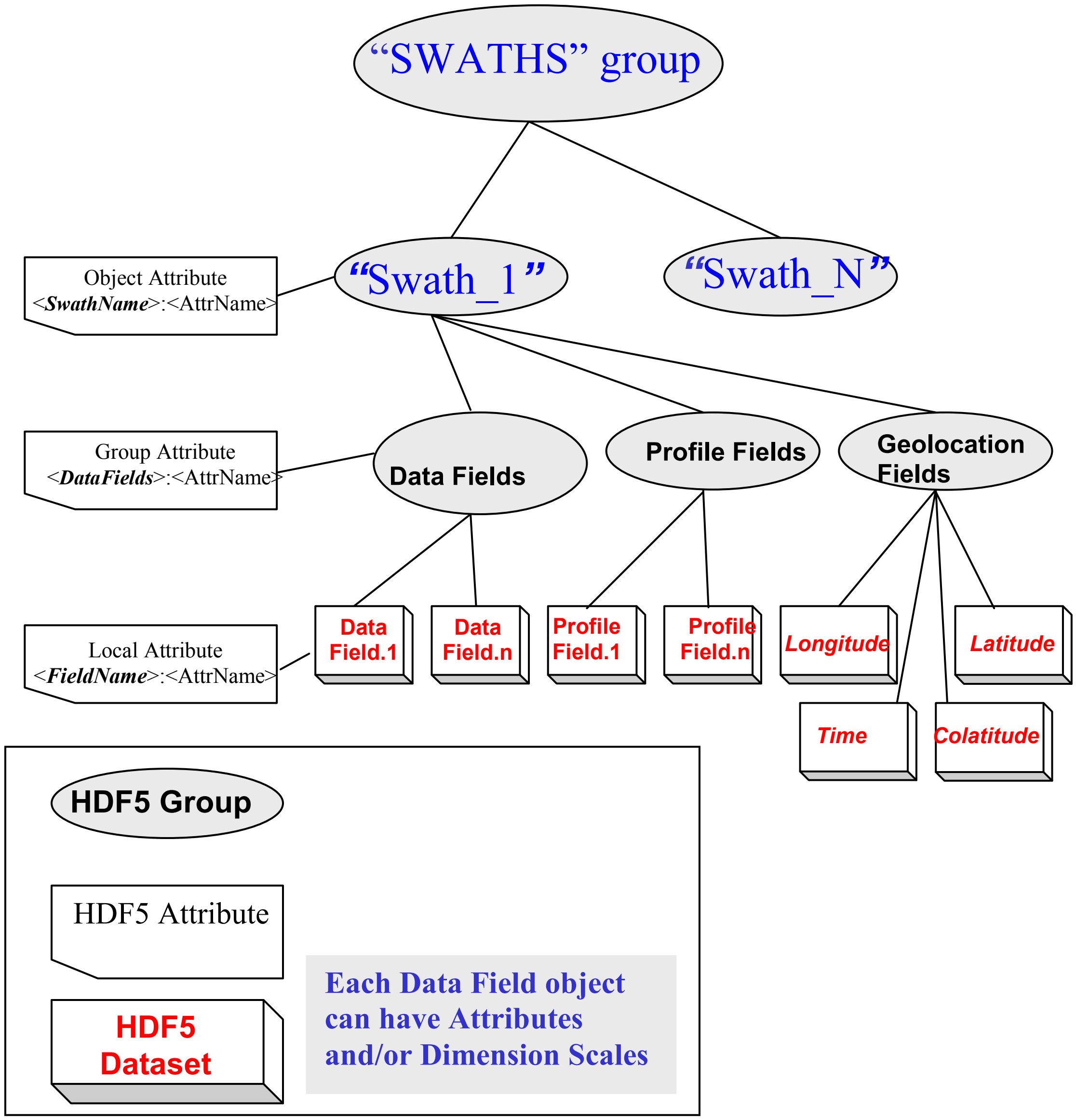

Figure 6.1-6 HDF5 Objects Created by an HDF-EOS5 Program for Swath Objects

6.2 GRID Data Model

As described in Section 6.1, Swaths carry geolocation information as a series of individually located points (tie points or ground control points). Grids, though, carry their geolocation in a much more compact form. A grid merely contains a set of projection equations (or references to them) along with their relevant parameters. Together, these relatively few pieces of information define the location of all points in the grid. The equations and parameters can then be used to compute the latitude and longitude for any point in the grid.

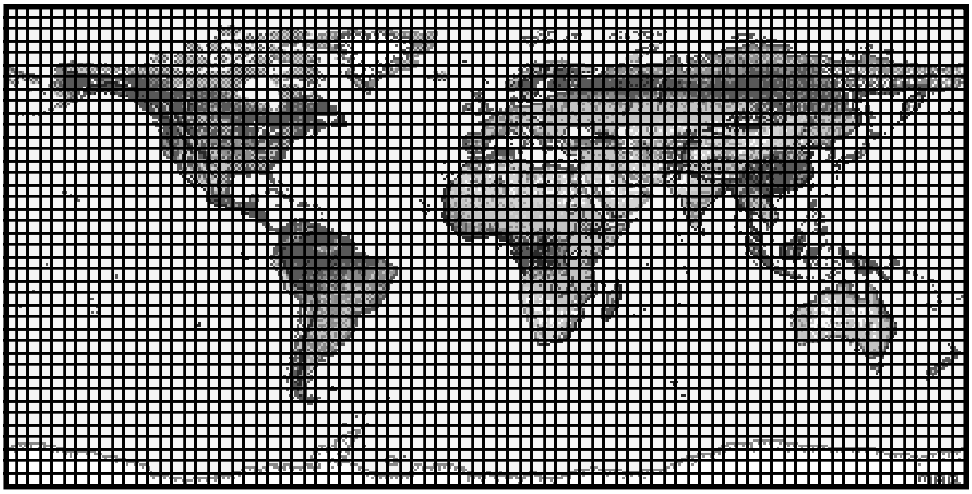

Figure 6.2-1. A Data Field in a Mercator-Projected Grid

In loose terms, each data field constitutes a map in a given standard projection. Although there may be many independent Grids in a single HDF-EOS file, within each Grid only one projection may be chosen for application to all data fields. Figures 6.2-1 and 6.2-2 show how a single data field may look in a Grid using two common projections.

There are three important features of a Grid data set: the data fields, the dimensions, and the projection. Each of these is discussed in detail in the following subsections.

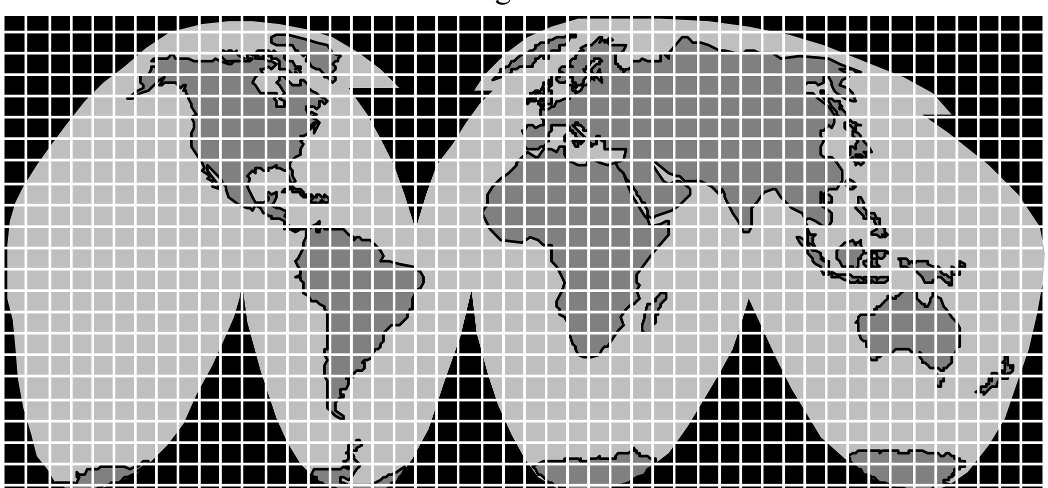

Figure 6.2-2. A Data Field in an Interrupted Goode's Homolosine-Projected Grid

6.2.1 Data Fields

The data fields are, of course, the most important part of the Grid. Data fields in a Grid data set are rectilinear arrays of two or more dimensions. Most commonly, they are simply two-dimensional rectangular arrays. Generally, each field contains data of similar scientific nature which must share the same data type. In general Grid supports all HDF5 supported datatypes. However, some Grid APIs, such as GD_interpolate, only support a few basic datatypes such as “short integer”, “integer”, “float”, and ‘Double”. The data fields are related to each other by common geolocation. That is, a single set of geolocation information is used for all data fields within one Grid data set.

6.2.2 Dimensions

Dimensions are used to relate data fields to each other and to the geolocation information. To be interpreted properly, each data field must make use of the two predefined dimensions: “XDim” and “YDim”. These two dimensions are defined when the grid is created and are used to refer to the X and Y dimensions of the chosen projection. Like for swath objects the grid dimensions are stored in the structure metadata. Although there is a limit of eight dimensions a data field in a Grid data set my have, it is not likely that many fields will need more than three: the predefined dimensions “XDim” and “YDim” and a third dimension for depth or height.

6.2.3 Projections

The projection is really the heart of the Grid structure. Without the use of a projection, the Grid would not be substantially different from a Swath. The projection provides a convenient way to encode geolocation information as a set of mathematical equations which are capable of transforming Earth coordinates (latitude and longitude) to X-Y coordinates on a sheet of paper.

The choice of a projection to be used for a Grid is a critical decision for a data product designer. There are a large number of projections that have been used throughout history. In fact, some projections date back to ancient Greece. Many projections are supported by the HDF-EOS API, including: Geographic, Universal Transverse Mercator, Albers Conical Equal Area, Lambert Conformal, Mercator, Polar Stereographic, Polyconic, Transverse Mercator, Lambert Azimuthal Equal Area, Hotin Oblique Mercator, Space Oblique, Interrupted Goode’s Homolosine, Integerized Sinusoidal, and Cylindrical Equal area.

The HDF-EOS5 API assumes that the data producer will use to create the data the General Coordinate Transformation Package (GCTP), a library of projection software available from the U.S. Geological Survey. The Grid interface allows the data producer to specify the exact GCTP parameters used to perform the projection and will provide for basic subsetting of the data fields by latitude/longitude bounding box.

See section 8.3 below for further details on the usage of the GCTP package.

6.2.4 HDF5 Objects in HDF-EOS 5 Grid Objects

Figure 6.2-3 shows the relationship between HDF-EOS5 Grid objects and HDF5 objects. As shown HDF-EOS5 creates a HDF5 group called “GRIDS” to hold all Grid objects. The Grid objects, which again are HDF5 groups with user defined names, contain “Data Fields” group and Grid related attributes called object attributes. The group “Data Fields” is the group that holds the user defined data field datasets and optional group attributes. In addition to the science data the datasets contain field attributes that are local to the field. A few examples of such attributes are units, fillvalue, etc. Figure 6.2-3 shows an example of a grid structure and a structure metadata associated with the grid. Again, as in swath all attributes are optional except the "_FillValue" attribute for the datasets which are created internally be HDF-EOS5 for every dataset and are assigned values other than zero when user sets the value using Grid's fillvalue setting routine.

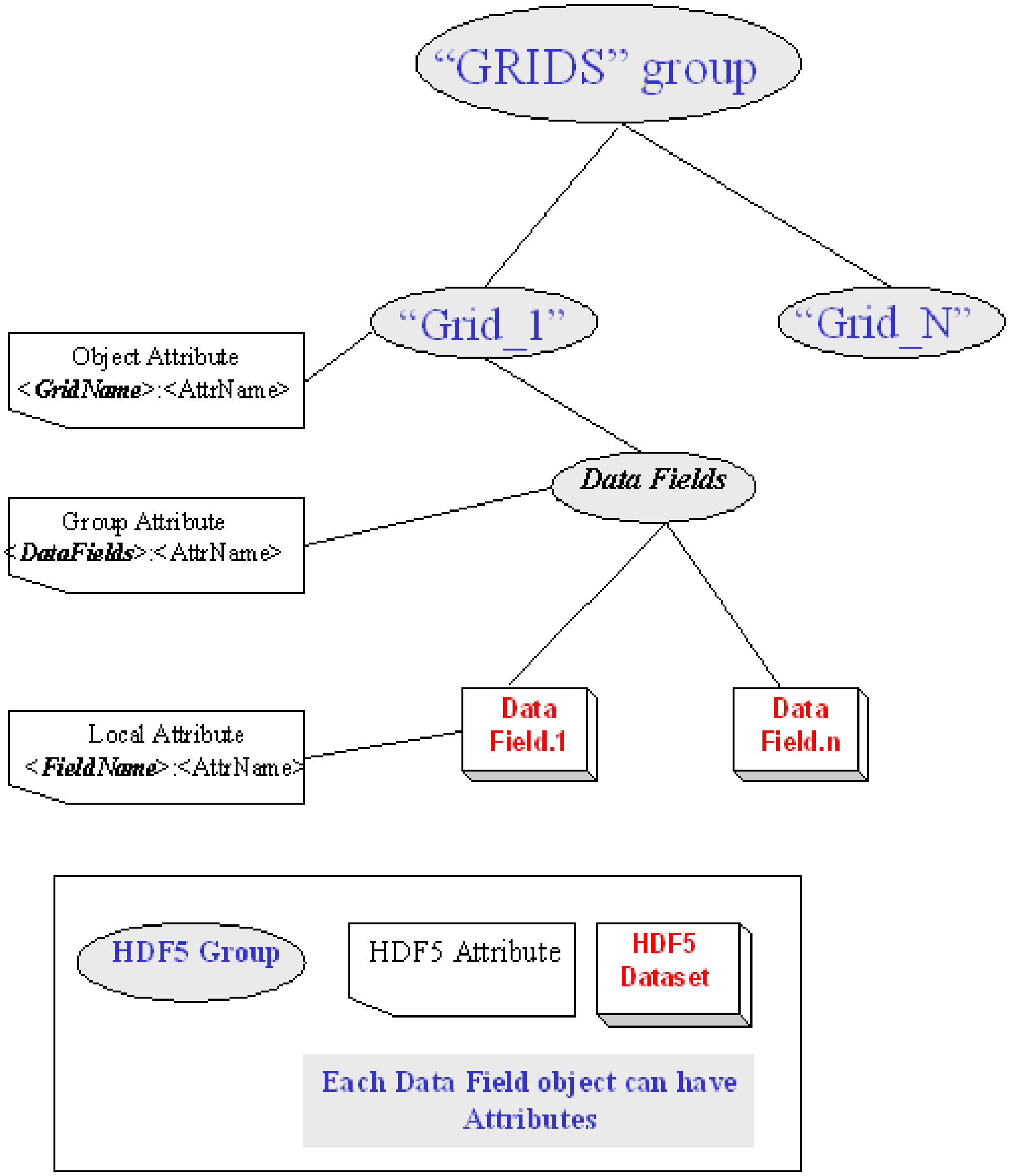

Figure 6.2-3 Grid objects created in an HDF-EOS5 file. ("GRIDS" and "Data Fields" groups are defined internally by HDF-EOS5).

Figure 6.2-5 shows a schematic of a grid structure containing two and three dimensional arrays. It also shows part of the related structure metadata stored in “StructMetadata” dataset shown in Figure 7.2-1.

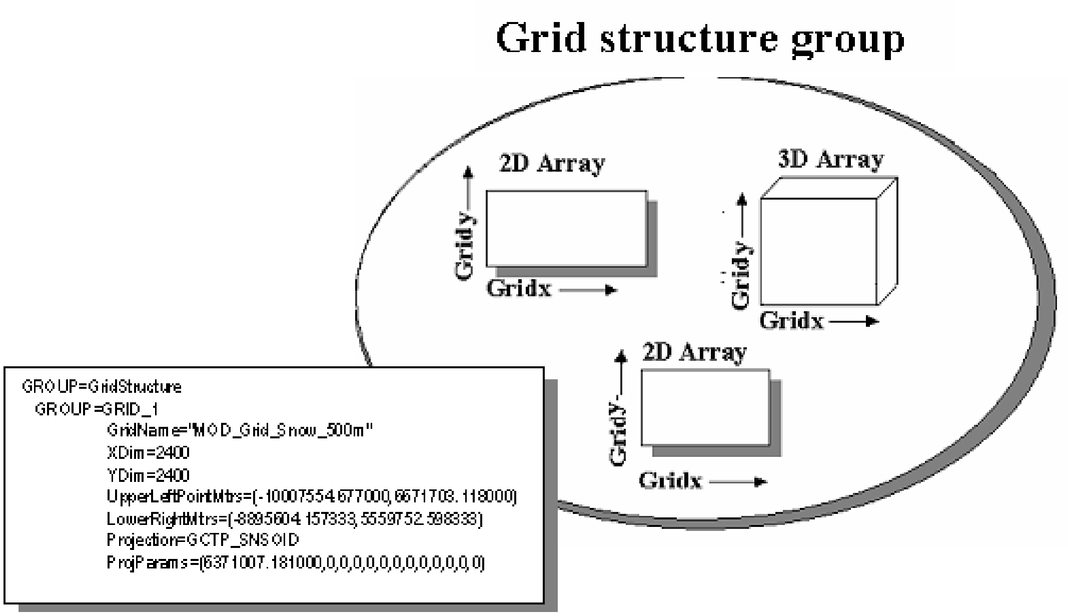

Figure 6.2-5 A schematic of a single grid structure containing two 2D arrays and one 3D array. The metadata describing the structure is contained inside the structural metadata text block.

7. HDF-EOS5 File Format

7.1 Introduction

In this Section, we present a brief introduction to the file format of HDF-EOS5. A detailed discussion, as well as an operational description of HDF-EOS5 can be found in HDF-EOS5 Interface Based on HDF5 Project (Volume 1 and Volume 2, 2005). HDF-EOS5 is composed of HDF5 objects. The file format of HDF-EOS, the ordering and meaning of bytes stored on disk or memory is therefore the same as the file format of HDF5. (see HDF5 User Documentation Release, U. of Illinois, Urbana Champaign, 2004)

7.2 HDF-EOS5 File Format

7.2.1 Overview

The HDF5 File Format defines the low-level objects in terms of a sequence of bytes. The HDF5 persistent objects are described in terms of the low-level objects, thus creating a mapping from the HDF5 data model to the set of byte sequences (HDF5 File Format, NCSA, U. of Illinois, Urbana-Champaign, 2004) HDF-EOS5 on the other hand maps HDF-EOS objects, structures, onto basic HDF5 objects such as Groups, Datasets, and Attributes. Therefore, in the following section we will see how HDF-EOS5 objects are constructed using HDF5 objects.

7.2.2 Structure of an HDF-EOS5 File

An HDF-EOS5 file is any valid HDF5 file (i.e., any file created by the NCSA HDF5 library), that contains HDF-EOS structures, e.g. Swath and Grid as described in Section 6. The existence of Swath or Grid structures in an HDF-EOS file implies the existence of another family of global attributes called "StructMetadata.X". The file can contain any number of Swath and/or Grid data structures. HDF-EOS5 can also contain a family of global attributes called "coremetadata.X", where ".X" is a sequence number beginning at 0 and running as high as 9. Optional data objects which may appear in an HDFEOS file include, another family of global attributes called "archivemetadata.X".

HDF-EOS5 related global attributes such as "StructMetadata" or "coremetadata" are written in a group called "HDFEOS INFORMATION". These attributes are basically either supplemental HDF5 objects, such as "HDFEOS Version" attribute in the "HDFEOS INFORMATION" group, or HDF5 datasets with ASCII contents. These global attributes provide information on the structure of an HDF-EOS file or information on the data granule that file contains. Other optional user-added global attributes such as "PGEVersion", "OrbitNumber", etc. are written as HDF5 attributes into a group called "FILE ATTRIBUTES" (see Figure 7.2-1). These attributes, written in the form of HDF5's supplemental attribute objects, usually provide quick reference to the origin/nature of the data. Please note that the "HDFEOS Version" attribute is created internally by HDF-EOS5 upon creating output HDF file.

7.2.3 Core Metadata

Core metadata represent information which is used to populate searchable database tables within the ECS archives. Data users use this information to locate particular HDF-EOS5 data granules. These metadata, which are defined in Release B-1 Earth Sciences Data Model, are also copied in the "coremetadata.X" (X= 0,...,n) family of global HDF-EOS 5 attributes within an HDF-EOS file. The syntax of these metadata is compliant with the Object Description Language (ODL)[http://pds.jpl.nasa.gov/documents/sr/Chapter12.pdf]. Tools for formatting, accessing and writing core metadata are provided in the EOS Science Data Processing (SDP) Toolkit. (SDP Toolkit Users Guide for the ECS Project). Note that Core metadata and the SDP Toolkit metadata tools are used for archival and distribution functions of EOS data systems. They are a separate standard, can be associated with the HDF-EOS 5 standard, but not necessarily part of the HDF-EOS 5 standard. HDF-EOS 5 file can be written without Core metadata, however, those files would not be assessable through the EOS archives.

7.2.4 Archive Metadata

Archive metadata represent information that, by definition, will not be searchable. It contains whatever information the file creator considers useful to be in the file, but which will not be directly accessible by

ECS databases. Archive metadata are also accessed via SDP Toolkit calls and are written in ODL syntax into the “archivemetadata.X”, (X=0,...,n) family of global attributes. (see SDP Toolkit Users Guide for the ECS Project).

7.2.5 Structural Metadata

Structural metadata describe the contents and structure of an HDF-EOS file. That is, these metadata describe how geolocation, temporal, projection information are to be associated with the data itself. Structural metadata are present in the file only if the HDF-EOS library has been invoked to create a Grid Swath, and Point structure. These metadata are stored in the “StructMetadata.X” family of global attributes and are created and maintained by the HDF-EOS library. They are also stored in ODL format.

The Structural metadata is internal to the HDF-EOS library and is not intended to be directly accessed by data producers or users. Therefore, all access to these metadata should be via appropriate function calls in the HDF-EOS library.

For reference purposes, structural metadata attributes and their definitions are itemized in Appendix A.

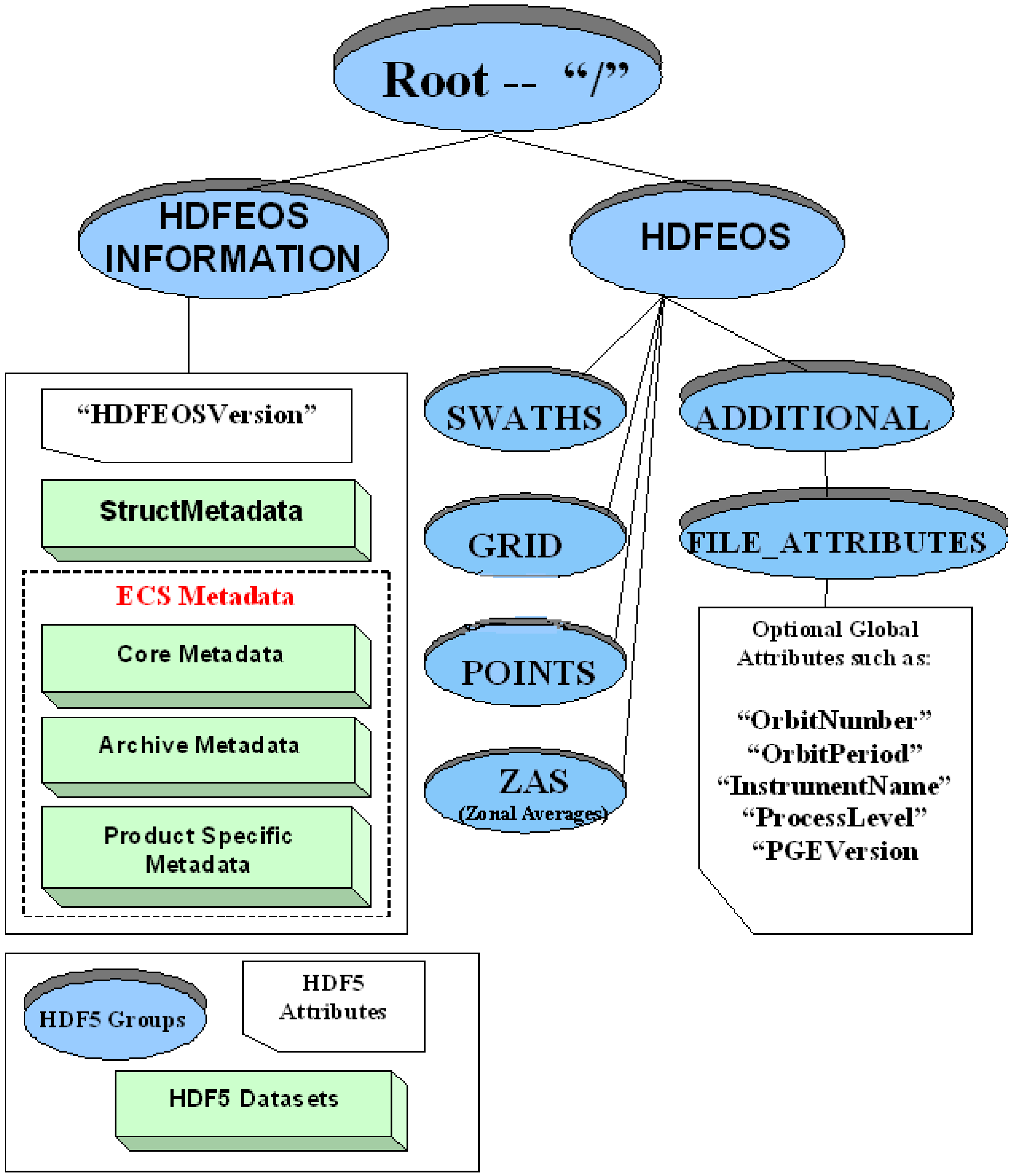

Figure 7.2-1 Top Level of HDF-EOS5 File

7.2.6 Swath Structure

Swath structures are implemented as a hierarchy of HDF5 groups containing a number of other HDF5 groups, datasets and/or HDF5's supplemental attribute objects. All groups, datasets and attributes that are part of any Swath structure carry the class “SWATH”. All one-dimensional and multi-dimensional fields are implemented as HDF5 datasets. The following limitations apply to Swath structures:

- The reserved field names for special purpose geolocation fields are “Longitude”, “Latitude”, “Colatitude”, and “Time” (case sensitive). These fields are subject to the following requirements:

Field Name	Data Type	Format
Longitude	float32 or float64	Decimal degrees on the range [-180.0, 180.0)
Latitude	float32 or float64	Decimal degrees on the range [-90.0, 90.0]
Colatitude	float32 or float64	Decimal degrees on the range [0.0, 180.0]
Time	float64	TAI93 (seconds until(-)/since(+) midnight, 1/1/93)

These fields may be one- or two-dimensional. The HDF-EOS library can check on the validity of geofield names and issue warnings if there are similarity between the user defined geofields and the reserved geofield names, to avoid possibility of using invalid uppercase or lowercase letters in the reserved geofield names (e.g. using “LATITUDE” instead of the reserved name “Latitude”).

Non-reserved fields may have up to 8 dimensions.

An “unlimited” dimension must be the first dimension (in C-order).

For all multi-dimensional fields in scan- or profile-oriented Swaths, the dimension representing the “along track” dimension must precede the dimension representing the scan or profile dimension(s) (in C-order)[2].

Compression is selectable at the field level within a Swath. All HDF5-supported compression methods are available through the HDF-EOS5 library. The compression method is stored within the file. Subsequent use of the library will un-compress the file. As in HDF5 the data needs to be chunked before the compression is applied.

Field names may be up to 64 characters in length.

Any character can be used with the exception of, ",", ";", " and "/".

Names are case sensitive.

Names must be unique within a particular Swath structure. Fields with identical names are allowed in different swaths of the same file, but they must be accessed using only HDF-EOS5 APIs. Otherwise, bypassing StructMetadata by using HDF5 APIs to access such fields may result in error.

[2] The “along track” dimension is the slowest of “along track” and “cross track” dimensions in both C and Fortran. Thus a C-order dimension list of “along track, cross track” for a geofield should have “cross track, along track ” order in Fortran. Similarly a C-order dimension list of “Band, DataTrack, DataXtrack” for a 3-D data field should have “DataXtrack, DataTrack, Band” order in Fortran.

7.2.7 Grid Structure

Grid structures are implemented as a hierarchy of HDF5 groups containing several datasets and attributes. All groups, datasets and attributes that are part of any Grid structure carry the class “GRID”. Each data field within a Grid structure is implemented as a single dataset. The following limitations apply to Grid structures:

Fields may have from 2 to 8 dimensions.

Compression is selectable at the field level within a Grid. All HDF5-supported compression methods are available through the HDF-EOS5 library. Table 7-1 shows all supported compression methods.

Table 7-1. Compression Methods

Compression Code	Value	Explanation
HE5_HDFE_COMP_NONE	0	No Compression
HE5_HDFE_COMP_RLE	1	Run Length Encoding Compression (not supported)
HE5_HDFE_COMP_NBIT	2	NBIT Compression
HE5_HDFE_COMP_SKPHUFF	3	Skipping Huffman (not supported)
HE5_HDFE_COMP_DEFLATE	4	gzip Compression
HE5_HDFE_COMP_SZIP_CHIP	5	szip Compression, Compression exactly as in hardware
HE5_HDFE_COMP_SZIP_K13	6	szip Compression, allowing k split = 13 Compression
HE5_HDFE_COMP_SZIP_EC	7	szip Compression, entropy coding method
HE5_HDFE_COMP_SZIP_NN	8	szip Compression, nearest neighbor coding method
HE5_HDFE_COMP_SZIP_K13orEC	9	szip Compression, allowing k split = 13 Compression, or entropy coding method
HE5_HDFE_COMP_SZIP_K13orNN	10	szip Compression, allowing k split = 13 Compression, or nearest neighbor coding method
HE5_HDFE_COMP_SHUF_DEFLATE	11	shuffling + deflate(gzip) Compression
HE5_HDFE_COMP_SHUF_SZIP_CHIP	12	shuffling + Compression exactly as in hardware
HE5_HDFE_COMP_SHUF_SZIP_K13	13	shuffling + allowing k split = 13 Compression
HE5_HDFE_COMP_SHUF_SZIP_EC	14	shuffling + entropy coding method
HE5_HDFE_COMP_SHUF_SZIP_NN	15	shuffling + nearest neighbor coding method
HE5_HDFE_COMP_SHUF_SZIP_K13orEC	16	shuffling + allowing k split = 13 Compression, or entropy coding method
HE5_HDFE_COMP_SHUF_SZIP_K13orNN	17	shuffling + allowing k split = 13 Compression, or nearest neighbor coding method
NOTE: For Compression the data storage must be CHUNKED first.		

The compression method is stored within the file. Subsequent use of the library will un-compress the file. The data should be chunked before compression is applied.

Field names may be up to 64 characters in length.

Any character can be used with the exception of, ",", ";", " and "/".

Names are case sensitive.

Names must be unique within a particular Grid structure. Fields with identical names are allowed in different grids of the same file, but must be accessed using only HDF-EOS5 APIs. Otherwise, bypassing StructMetadata by using HDF5 APIs to access such fields may result in error.

7.2.8 Point Structure

The Point structures are implemented as a hierarchy of HDF5 groups each containing a Data group and a Linkage group and attributes.

The Data group contains a series of data records taken at [possibly] irregular time intervals and at scattered geographic locations. They are loosely organized form of geolocated data that are supported by HDF-EOS. The data records are hierarchically arranged to include up to seven indexing levels (a total 0f eight levels, including the bottom level data table). Levels are linked by a common field name called *LinkField.* A level can contain any number of fields and records. The order in which the levels are defined determines the (0-based) level index.

The Linkage group contains two tables showing possible forward and backward linkage between the levels. Usually shared info is stored in Parent level while data values are stored in Child level. Data and Linkage groups are created automatically when the level is defined.

The following limitations apply to Point structures:

The level name should not contain slashes ("/") and may be up to 256 characters in length.

Names are case sensitive.

The values for the *LinkField* in the Parent level must be unique.

The records in Child level is not monotonic in *LinkField*, otherwise FWDPOINTER Linkage will not be set (actually first one is set to (-1,-1) to indicate problem with FWDPOINTER).

7.2.9 Zonal Average (ZA) Structure

The Zonal Average structure is basically a swath like structure without geolocation. The interface is designed to support data that has not associated with specific geolocation information.

7.2.10 Hybrid HDF-EOS5 and HDF Files

An HDF-EOS file can contain any number of Grid, Point, Swath, Zonal Average, and Profile data structures. Unlike the HDF-EOS2 files which have two Gigabyte size limits, HDF-EOS5 file has no size limits. An HDF-EOS5 file can also contain plain HDF5 objects for special purposes. HDF5 objects must be accessed by the HDF5 library and not by HDFEOS5 extensions. A user should note however, that inclusion of HDF5 objects will require more knowledge of file contents on the part of an applications developer or data user. A user should also note that HDF5 is a directory structure and that a file containing 1000's of objects could cause program execution slow-downs.

8. HDF-EOS 5 Library/ Programming Model

8.1 The Swath Data Interface

The SW interface consists of routines for storing, retrieving, and manipulating data in swath data sets.

8.1.1 SW API Routines

All C routine names in the swath data interface have the prefix "HE5_SW" and the equivalent FORTRAN routine names are prefixed by "he5_sw." The SW routines are classified into the following categories:

- *Access routines* initialize and terminate access to the SW interface and swath objects (including opening and closing files).
- *Definition* routines allow the user to set key features of a swath objects.
- *Basic I/O* routines read and write data and metadata to a swath objects.
- *Inquiry* routines return information about data contained in a swath objects
- *Subset* routines allow reading of data from a specified geographic region.

The SW function calls are listed in Table 8-1 and are described in detail in the 2nd volume of HDF-EOS5 Users Guide (HDF-EOS Interface Based on HDF5, Volume 2: Function Reference Guide, Technical Paper, 175-EMD-002 Revision 03, April 2005).

Table 8-1. Summary of the Swath Interface (1 of 3)

Category	Routine Name		Description
	C	FORTRAN	
Access	HE5_SWopen	he5_swopen	Opens or creates HDF file in order to create, read, or write a swath
	HE5_SWcreate	he5_swcreate	Creates a swath within the file
	HE5_SWattach	he5_swattach	Attaches to an existing swath within the file
	HE5_SWdetach	he5_swdetach	Detaches from swath interface
	HE5_SWclose	he5_swclose	Closes file
Definition	HE5_SWdefdim	he5_swdefdim	Defines a new dimension within the swath
	HE5_SWdefdimmap	he5_swdefmap	Defines the mapping between the geolocation and data dimensions
	HE5_SWdefidxmap	he5_swdefimap	Defines a non-regular mapping between the geolocation and data dimension
	HE5_SWdefgeofield	he5_swdefgfld	Defines a new geolocation field within the swath
	HE5_SWdefdatafield	he5_swdefdfld	Defines a new data field within the swath
	HE5_SWdefcomp	he5_swdefcomp	Defines a field compression scheme
	HE5_SWdefchunk	he5_swdefchunk	Define chunking parameters
	HE5_SWdefcomchur	he5_swdefcomch	Defines compression with automatic chunking
	HE5_SWsetalias	he5_swsetalias	Defines alias for data field
	HE5_SWdropalias	he5_swdrpalias	Removes alias from the list of field aliases
	HE5_SWfldrename	he5_swfldrnm	Changes the field name
Basic I/O	HE5_SWwritefield	he5_swwrfld	Writes data to a swath field
	HE5_SWwritegeome	he5_swwrgmeta	Writes field metadata for an existing swath geolocation field
	HE5_SWwritedatame	he5_swwrdmeta	Writes field metadata for an existing swath data field
	HE5_SWreadfield	he5_swrdfld	Reads data from a swath field.
	HE5_SWwriteattr	he5_swwrattr	Writes/updates attribute in a swath
	HE5_SWreadattr	he5_swrdattr	Reads attribute from a swath
	HE5_SWwritegeogrp	he5_swwrgeogattr	Writes/updates group Geolocation Fields attribute in a swath
	HE5_SWwritegrpattr	he5_swwrgattr	Writes/updates group Data Fields attribute in a swath
	HE5_SWwritelocattr	he5_swwrlattr	Write/updates local attribute in a swath
	HE5_SWreadgeogrp	he5_swrdgeogattr	Reads attribute in Geolocation Fields from swath
	HE5_SWreadgrpattr	he5_swrdgattr	Reads attribute in Data Fields from a swath
	HE5_SWreadlocattr	he5_swrdlattr	Reads attribute from a swath
	HE5_SWsetfillvalue	he5_swsetfill	Sets fill value for the specified field
	HE5_SWgetfillvalue	he5_swgetfill	Retrieves fill value for the specified field
Inquiry	HE5_SWaliasinfo	he5_swaliasinfo	Retrieves information about field aliases
	HE5_SWinqdims	he5_swinqdims	Retrieves information about dimensions defined in swath
	HE5_SWinqmaps	he5_swinqmaps	Retrieves information about the geolocation relations defined
	HE5_SWinqidxmaps	he5_swinqimaps	Retrieves information about the indexed geolocation/data mappings defined
	HE5_SWinqgeofields	he5_swinqgflds	Retrieves information about the geolocation fields defined
	HE5_SWinqdatafield	he5_swinqdflds	Retrieves information about the data fields defined
	HE5_SWinqattrs	he5_swinqattrs	Retrieves number and names of attributes defined

Table 8-1. Summary of the Swath Interface (2 of 3)

Category	Routine Name C	Routine Name FORTRAN	Description
Inquiry	HE5_SWinqdatatype	he5_swidtype	Returns data type information about specified fields in swath
	HE5_SWinqdfldalias	he5_swinqdfldalias	Returns information about data fields & aliases defined in swath
	HE5_SWinqgfldalias	he5_swinqgfldalias	Returns information about geolocation fields & aliases defined in swath
	HE5_SWinqgeogrpattrs	he5_swinqgeogattrs	Retrieve information about group Geolocation Fields attributes defined in swath
	HE5_SWinqgrpattrs	he5_swinqgattrs	Retrieve information about group Data Fields attributes defined in swath
	HE5_SWinqlocattrs	he5_swinqlattrs	Retrieve information about local attributes defined in swath
	HE5_SWlocattrinfo	he5_swlocattrinfo	Returns information about a data field's local attribute(s)
	HE5_SWnentries	he5_swnentries	Returns number of entries and descriptive string buffer size for a specified entity
	HE5_SWdiminfo	he5_swdiminfo	Retrieve size of specified dimension
	HE5_SWchunkinfo	he5_swchunkinfo	Retrieve chunking information
	HE5_SWmapinfo	he5_swmapinfo	Retrieve offset and increment of specified geolocation mapping
	HE5_SWidxmapinfo	he5_swimapinfo	Retrieve offset and increment of specified geolocation mapping
	HE5_SWattrinfo	he5_swattrinfo	Returns information about swath attributes
	HE5_SWgeogrpattrinfo	he5_swgeogattrinfo	Returns information about group Geolocation Fields attribute
	HE5_SWgrpattrinfo	he5_swgattrinfo	Returns information about group Data Fields attribute
	HE5_SWfieldinfo	he5_swfldinfo	Retrieve information about a specific geolocation or data field
	HE5_SWcompinfo	he5_swcompinfo	Retrieve compression information about a field
	HE5_SWinqswath	he5_swinqswath	Retrieves number and names of swaths in file
	HE5_SWregionindex	he5_swregidx	Returns information about the swath region ID
	HE5_SWupdateidxmap	he5_swupimap	Update map index for a specified region
	HE5_SWgeomapinfo	he5_swgmapinfo	Retrieve type of dimension mapping for a dimension
Subset	HE5_SWdefboxregion	he5_swdefboxreg	Define region of interest by latitude/longitude
	HE5_SWregioninfo	he5_swreginfo	Returns information about defined region
	HE5_SWextractregion	he5_swextreg	Read a region of interest from a field
	HE5_SWdeftimeperiod	he5_swdeftmeper	Define a time period of interest
	HE5_SWperiodinfo	he5_swperinfo	Returns information about a defined time period
	HE5_SWextractperiod	he5_swextper	Extract a defined time period
	HE5_SWdefvrtregion	he5_swdefvrtreg	Define a region of interest by vertical field
	HE5_SWindexinfo	he5_swindexinfo	Returns the indices about a subsetted region
	HE5_SWdupregion	he5_swdupreg	Duplicate a region or time period
Profile	HE5_PRdefine	he5_prdefine	Defines profile data structure
	HE5_PRread	he5_prread	Reads profile data
	HE5_PRwrite	he5_prwrite	Writes profile data
	HE5_PRinquire	he5_prinquire	Retrieves information about profiles

	HE5_PRinfo	he5_prinfo	Return information about profile

Table 8-1. Summary of the Swath Interface (3 of 3)

Category	Routine Name: C	Routine Name: FORTRAN	Description
Profile	HE5_PRreclaimspace	Not available	Reclaims memory used by data buffer in HE5_PRread()call
	HE5_PRwritegrpattr	he5_prwrgattr	Writes/updates group Profile Fields attribute in a swath
	HE5_PRreadgrpattr	he5_prrdgattr	Reads attribute in group Profile Fields from a swath
	HE5_PRinqgrpattrs	he5_prinqgattrs	Retrieves information about group Profile Fields attributes defined in swath
	HE5_PRgrpattrinfo	he5_prgattrinfo	Returns information about a group Profile Fields attribute
External Files	HE5_SWmountexternal	Not available	Mount external data file
	HE5_SWreadexternal	Not available	Read external data set
	HE5_SWunmount	Not available	Dismount external data file
External Data Sets	HE5_SWsetextdata	he5_swsetxdat	Set external data set
	HE5_SWgetextdata	he5_swgetxdat	Get external data set

8.1.2 File Identifiers

As with all HDF-EOS5 interfaces, file identifiers in the HE5_SW interface are of hid_t HDF5 type, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces such as those created by HE5_GD interface.

8.1.3 Swath Identifiers

Before a swath data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a *swath identifier*. After a swath data set has been opened for access, it is uniquely identified by its swath identifier.

8.1.4 Programming Model

The programming model for accessing a swath data set through the HE5_SW interface is as follows:

1. Open the file and initialize the HE5_SW interface by obtaining a Swath Interface identifier from a file name.
2. Open or create a swath object by obtaining a swath identifier from a swath name.
3. Perform desired operations on the data set.
4. Close the swath data set by disposing of the swath identifier.
5. Terminate swath access to the file by disposing of the Swath Interface identifier.

The following is a code fragment illustrating the programming model. Appendix B shows the contents of the output HDF-EOS5 files containing Swath objects. The file is the result of applying h5dump on the hdf-eos5 output.

```
/* In this example we open an HDF-EOS file, (2) create the swath
   object within the file, and define the swath field dimensions.
```

```
Open a new HDF-EOS swath file, "Swath.he5". Assuming that this
file may not exist, we are using "H5F_ACC_TRUNC" access code.
The "HE5_SWopen" function returns the swath file ID, swfid,
which is used to identify the file in subsequent calls to the
HDF-EOS library functions. */

swfid = HE5_SWopen("Swath.he5", H5F_ACC_TRUNC);

/* Create the swath, "Swath1", within the file */
SWid = HE5_SWcreate(swfid, "Swath1");
SWid_index = HE5_SWcreate(swfid, "Swath2");

/* Define dimensions and specify their sizes */
status = HE5_SWdefdim(SWid, "GeoTrack", 20);
status = HE5_SWdefdim(SWid, "GeoXtrack", 10);
status = HE5_SWdefdim(SWid, "Res2tr", 40);
status = HE5_SWdefdim(SWid, "Res2xtr", 20);
status = HE5_SWdefdim(SWid, "Bands", 15);
status = HE5_SWdefdim(SWid, "ProfDim", 4);

/* Define "Unlimited" Dimension  */
status = HE5_SWdefdim(SWid, "Unlim", H5S_UNLIMITED);

/* Once the dimensions are defined, the relationship (mapping) between the
geolocation dimensions, such as track and cross track, and the data
dimensions, must be established. This is done through the "HE5_SWdefdimmap"
function. It takes as input the swath id, the names of the dimensions
designating the geolocation and data dimensions, respectively, and the
offset and increment defining the relation.

In the first example we relate the "GeoTrack" and "Res2tr" dimensions
with an offset of 0 and an increment of 2.  Thus the ith element of
"Geotrack" corresponds to the 2 * ith element of "Res2tr".

In the second example, the ith element of "GeoXtrack" corresponds to the
2 * ith + 1 element of "Res2xtr".

Note that there is no relationship between the geolocation dimensions
and the "Bands" dimension. */

/* Define Dimension Mappings  */
status = HE5_SWdefdimmap(SWid, "GeoTrack", "Res2tr", 0, 2);

status = HE5_SWdefdimmap(SWid, "GeoXtrack", "Res2xtr", 1, 2);

/* Define Indexed Mapping  */
 IndexMap[0] = 1L;
 IndexMap[1] = 5L;
 IndexMap[2] = 8L;
 IndexMap[3] = 12L;
 IndexMap[4] = 17L;
 IndexMap[5] = 20L;

status = HE5_SWdefdim(SWid_index, "Res2tr_indexed", 40);
status = HE5_SWdefdim(SWid_index, "IndexTrack", 6);
```

```
status = HE5_SWdefidxmap(SWid_index, "IndxTrack", "Res2tr_indexed ", indx);

/* Create Geofield “Time” and Datafield “Temperature” */
status = HE5_SWdefgeofield(SWid, "Time", "GeoTrack", NULL,
                           H5T_NATIVE_DOUBLE, 0);

fillvalue     = -999.0;
status = HE5_SWsetfillvalue(SWid, "Temperature", H5T_NATIVE_FLOAT,
                            &fillvalue);
status = HE5_SWdefdatafield(SWid, "Temperature", "Res2tr,Res2xtr",
                            NULL,H5T_NATIVE_DOUBLE , 0);

charcount[0] = 13;
status = HE5_SWwritelocattr(SWid, "Temperature", "Unit",
                            H5T_NATIVE_CHAR,charcount,"Degree Kelvin");

/* Close the swath interface */
status = HE5_SWdetach(SWid);
status = HE5_SWdetach(SWid_index);

/* Close the swath file */
status = HE5_SWclose(swfid);
```

8.2 The Grid Data Interface

The GD interface consists of routines for storing, retrieving, and manipulating data in grid data sets.

8.2.1 GD API Routines

All C routine names in the grid data interface have the prefix "HE5_GD" and the equivalent FORTRAN routine names are prefixed by "he5_gd." The GD routines are classified into the following categories:

- *Access routines* initialize and terminate access to the GD interface and grid data sets (including opening and closing files).
- *Definition* routines allow the user to set key features of a grid data set.
- *Basic I/O* routines read and write data and metadata to a grid data set.
- *Inquiry* routines return information about data contained in a grid data set.
- *Subset* routines allow reading of data from a specified geographic region.

The GD function calls are listed in Table 8-2 and are described in detail in the Software Reference Guide that accompanies this document.

Table 8-2. Summary of the Grid Interface (1 of 2)

Category	Routine Name		Description
	C	FORTRAN	
Access	HE5_GDopen	he5_gdopen	Creates a new file or opens an existing one
	HE5_GDcreate	he5_gdcreate	Creates a new grid in the file
	HE5_GDattach	he5_gdattach	Attaches to a grid
	HE5_GDdetach	he5_gddetach	Detaches from grid interface
	HE5_GDclose	he5_gdclose	Closes file
Definition	HE5_GDdeforigin	he5_gddeforigin	Defines origin of grid pixel
	HE5_GDdefdim	he5_gddefdim	Defines dimensions for a grid
	HE5_GDdefproj	he5_gddefproj	Defines projection of grid
	HE5_GDdefpixreg	he5_gddefpixreg	Defines pixel registration within grid cell
	HE5_GDdeffield	he5_gddeffld	Defines data fields to be stored in a grid
	HE5_GDdefcomp	he5_gddefcomp	Defines a field compression scheme
	HE5_GDblkSOMoffset	None	This is a special function for SOM MISR data. Write block SOM offset values.
	HE5_GDdefcomtile	he5_gddefcomtle	Defines compression with automatic tiling
	HE5_GDsetalias	he5_gdsetalias	Defines alias for data field
	HE5_GDdropalias	he5_gddrpalias	Removes alias from a list of field aliases
Basic I/O	HE5_GDwritefieldmeta	he5_gdwrmeta	Writes metadata for field already existing in file
	HE5_GDwritefield	he5_gdwrfld	Writes data to a grid field.
	HE5_GDreadfield	he5_gdrdfld	Reads data from a grid field
	HE5_GDwriteattr	he5_gdwrattr	Writes/updates attribute in a grid.
	HE5_GDwritelocattr	he5_gdwrlattr	Writes/updates local attribute in a grid
	HE5_GDwritegrpattr	he5_gdwrgattr	Writes/updates group attribute in a grid
	HE5_GDreadattr	he5_gdrdattr	Reads attribute from a grid
	HE5_GDreadgrpattr	he5_gdrdgattr	Reads group attribute from a grid
	HE5_GDreadlocattr	he5_gdrdlattr	Reads local attribute from a grid

	HE5_GDsetfillvalue	he5_gdsetfill	Sets fill value for the specified field
	HE5_GDgetfillvalue	he5_gdgetfill	Retrieves fill value for the specified field
Inquiry	HE5_GDinqdims	he5_gdinqdims	Retrieves information about dimensions defined in grid
	HE5_GDinqfields	he5_gdinqdflds	Retrieves information about the data fields defined in grid
	HE5_GDinqattrs	he5_gdinqattrs	Retrieves number and names of attributes defined
	HE5_GDinqlocattrs	he5_gdinqlattrs	Retrieves information about local attributes defined for a field
	HE5_GDinqgrpattrs	he5_gdinqgattrs	Retrieves information about group attributes defined in grid
	HE5_GDnentries	he5_gdnentries	Returns number of entries and descriptive string buffer size for a specified entity
	HE5_GDaliasinfo	he5_gdaliasinfo	Retrieves information about aliases

Table 8-2. Summary of the Grid Interface (2 of 2)

Category	Routine Name		Description
	C	FORTRAN	
Inquiry	HE5_GDgridinfo	he5_gdgridinfo	Returns dimensions of grid and X-Y coordinates of corners
	HE5_GDprojinfo	he5_gdprojinfo	Returns all GCTP projection information
	HE5_GDdiminfo	he5_gddiminfo	Retrieves size of specified dimension.
	HE5_GDcompinfo	he5_gdcompinfo	Retrieves compression information about a field
	HE5_GDfieldinfo	he5_gdfldinfo	Retrieves information about a specific field in the grid
	HE5_GDinqgrid	he5_gdinqgrid	Retrieves number and names of grids in file
	HE5_GDinqfldalias	he5_gdinqfldalias	Returns information about data fields & aliases defined in grid
	HE5_GDinqdatatype	he5_gdinqdatatype	Returns data type information about specified fields in grid
	HE5_GDattrinfo	he5_gdattrinfo	Returns information about grid attributes
	HE5_GDgrpattrinfo	he5_gdgattrinfo	Returns information about a grid group attribute
	HE5_GDlocattrinfo	he5_gdlattrinfo	Returns information about a Data Field's local attribute(s)
	HE5_GDorigininfo	he5_gdorginfo	Returns information about grid pixel origin
	HE5_GDpixreginfo	he5_gdpreginfo	Returns pixel registration information for given grid
Subset	HE5_GDdefboxregion	he5_gddefboxreg	Defines region of interest by latitude/longitude
	HE5_GDregioninfo	he5_gdreginfo	Returns information about a defined region
	HE5_GDextractregion	he5_gdextrreg	Read a region of interest from a field
	HE5_GDdeftimeperiod	he5_gddeftmeper	Define a time period of interest
	HE5_GDdefvrtregion	he5_gddefvrtreg	Define a region of interest by vertical field
	HE5_GDgetpixels	he5_gdgetpix	Get row/columns for lon/lat pairs
	HE5_GDgetpixvalues	he5_gdgetpixval	Get field values for specified pixels
	HE5_GDinterpolate	he5_gdinterpolate	Perform bilinear interpolation on a grid field
	HE5_GDdupregion	he5_gddupreg	Duplicate a region or time period
Tiling	HE5_GDdeftile	he5_gddeftle	Define a tiling scheme
	HE5_GDtileinfo	he5_gdtileinfo	Retrieve tiling information
Utility	HE5_GDrs2ll	he5_gdrs2ll	Convert (r,s) coordinates to (lon,lat) for EASE grid
External Data Sets	HE5_GDsetextdata	he5_gdsetxdat	Set external data set
	HE5_GDgetextdata	he5_gdgetxdat	Get external data set

8.2.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the GD interface are of hid_t HDF5 type, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

8.2.3 Grid Identifiers

Before a grid data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a *grid identifier*. After a grid data set has been opened for access, it is uniquely identified by its grid identifier.

8.2.4 Programming Model

The programming model for accessing a grid object through the GD interface is as follows:

1. Open the file and initialize the GD interface by obtaining a file ID from a file name.
2. Open OR create a grid object by obtaining a grid ID from a grid name.
3. Perform desired operations on the data set.
4. Close the grid object by disposing of the grid ID.
5. Terminate grid access to the file by disposing of the file ID.

In this example we open the HDF-EOS grid file, "Grid.he5". Assuming that this file may not exist, we are using the H5F_ACC_TRUNC access code. The "HE5_GDopen" function returns the grid file ID, gdfid which is used to identify the file in subsequent calls to the HDF-EOS library functions. Appendix C shows how HDF-EOS 5 Grid objects are related to HDF objects.

```
gdfid = HE5_GDopen("Grid.he5", H5F_ACC_TRUNC);

xdim      = 5;
ydim      = 7;
zonecode   = 0;
spherecode = 0;
projparm = (double *)calloc( 16, sizeof(double) );
for (i = 0; i < 16; i++)
    {
    projparm[i] = 0.0;
    }
projparm[ 2 ]   = 0.9996;
projparm[ 4 ]   = -75000000.00;
projparm[ 6 ]   = 5000000.00;
uplft[0]  = 4855670.77539;
uplft[1]  = 9458558.92483;

lowrgt[0] = 5201746.43983;
lowrgt[1] = -10466077.24942;

/* Create "TM" Grid
Use default spheriod (Clarke 1866 - spherecode = 0) */
GDid   = HE5_GDcreate(gdfid, "TMGrid", xdim, ydim, uplft, lowrgt);

/* Define projection */
status = HE5_GDdefproj(GDid, HE5_GCTP_UTM, zonecode, spherecode,projparm);

/* Define "Time" Dimension */
status = HE5_GDdefdim(GDid, "Time", 10);
```

```
/* Define "Unlimited" Dimension */
status = HE5_GDdefdim(GDid, "Unlim", H5S_UNLIMITED);

/*NOTE:  This call should always precede the call to GDdeffield()*/
status = HE5_GDsetfillvalue(GDid, "Voltage", H5T_NATIVE_FLOAT,
                            &fillvalue);

status = HE5_GDdeffield(GDid,"Voltage","XDim,YDim",NULL,H5T_NATIVE_FLOAT,
                        0);


/* Close the grid interface */
status = HE5_GDdetach(GDid);

/* Close the grid file */
status = HE5_GDclose(gdfid);
```

To access several files at the same time, a calling program must obtain a separate ID for each file to be opened. Similarly, to access more than one grid object, a calling program must obtain a separate grid ID for each object. For example, to open two objects stored in two files, a program would execute the following series of C function calls:

```
gdfid_1 = HE5_GDopen(filename_1, access_mode);
gdid_1 = HE5_GDattach(gdfid_1, grid_name_1);
gdfid_2 = HE5_GDopen(filename_2, access_mode);
gdid_2 = HE5_GDattach(gdfid_2, grid_name_2);
<Optional operations>
status = HE5_GDdetach(gdid_1);
status = HE5_GDclose(gdfid_1);
status = HE5_GDdetach(gdid_2);
status = HE5_GDclose(gdfid_2);
```

Because each file and grid object is assigned its own identifier, the order in which files and objects are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.

8.3 GCTP Usage

The HDF-EOS Grid API uses the U.S. Geological Survey General Cartographic Transformation Package (GCTP) to define and subset grid structures. This section describes codes used by the package.

8.3.1 GCTP Projection Codes

HDF-EOS defines a unique code for any supported projection. These codes, defined as GCTP_<short projection name>, are assigned numbers 0 to 99. Once the projection is defined the projection code along with the projection parameters are written to the structure metadata (see Figure 6.2-4). HDF-EOS internally maps projection codes to the assigned numbers.

The following GCTP projections are supported for HDFEOS. The projection codes are used in the grid API described above:

```
GCTP_GEO          (0)   Geographic
GCTP_UTM          (1)   Universal Transverse Mercator
GCTP_ALBERS       (3)   Albers Conical Equal Area
GCTP_LAMCC        (4)   Lambert Conformal Conic
GCTP_MERCAT       (5)   Mercator
GCTP_PS           (6)   Polar Stereographic
GCTP_POLYC        (7)   Polyconic
GCTP_TM           (9)   Transverse Mercator
GCTP_LAMAZ        (11)  Lambert Azimuthal Equal Area
GCTP_HOM          (20)  Hotin Oblique Mercator
GCTP_SOM          (22)  Space Oblique Mercator
GCTP_GOOD         (24)  Interrupted Goode Homolosine
GCTP_ISINUS1      (31)  Integerized Sinusoidal Projection*
GCTP_ISINUS       (99)  Integerized Sinusoidal Projection*
GCTP_CEA          (97)  Cylindrical Equal-Area (for EASE grid with corners
                        in meters)**
GCTP_BCEA         (98)  Cylindrical Equal-Area (for EASE grid with grid corners
                        in packed degrees, DMS)**
```

* The Integerized Sinusoidal Projection was not part of the original GCTP package. It has been added by ECS. See *Level-3 SeaWiFS Data Products: Spatial and Temporal Binning Algorithms*. Additional references are provided in Section 2.

** The Cylindrical Equal-Area Projection was not part of the original GCTP package. It has been added by ECS. See Notes for section 8.3.4.

In the new GCTP package the Integerized Sinusoidal Projection is included as the 31st projection. The Code 31 was added to HDFEOS for users who wish to use 31 instead of 99 for Integerized Sinusoidal Projection.

Note that other projections supported by GCTP will be adapted for future HDF-EOS Versions as new user requirements are surfaced. For further details on the GCTP projection package, please refer to Section 8.3.5 and Appendix G of the SDP Toolkit Users Guide for the ECS Project, April, 2005, (333-EMD-001, Revision 03).

8.3.2 UTM Zone Codes

The Universal Transverse Mercator (UTM) Coordinate System uses zone codes instead of specific projection parameters. The table that follows lists UTM zone codes as used by GCTP Projection Transformation Package. C.M. is Central Meridian

Zone	C.M.	Range	Zone	C.M.	Range
01	177W	180W-174W	31	003E	000E-006E
02	171W	174W-168W	32	009E	006E-012E
03	165W	168W-162W	33	015E	012E-018E
04	159W	162W-156W	34	021E	018E-024E
05	153W	156W-150W	35	027E	024E-030E
06	147W	150W-144W	36	033E	030E-036E
07	141W	144W-138W	37	039E	036E-042E
08	135W	138W-132W	38	045E	042E-048E

09	129W	132W-126W	39	051E	048E-054E
10	123W	126W-120W	40	057E	054E-060E
11	117W	120W-114W	41	063E	060E-066E
12	111W	114W-108W	42	069E	066E-072E
13	105W	108W-102W	43	075E	072E-078E
14	099W	102W-096W	44	081E	078E-084E
15	093W	096W-090W	45	087E	084E-090E
16	087W	090W-084W	46	093E	090E-096E
17	081W	084W-078W	47	099E	096E-102E
18	075W	078W-072W	48	105E	102E-108E
19	069W	072W-066W	49	111E	108E-114E
20	063W	066W-060W	50	117E	114E-120E
21	057W	060W-054W	51	123E	120E-126E
22	051W	054W-048W	52	129E	126E-132E
23	045W	048W-042W	53	135E	132E-138E
24	039W	042W-036W	54	141E	138E-144E
25	033W	036W-030W	55	147E	144E-150E
26	027W	030W-024W	56	153E	150E-156E
27	021W	024W-018W	57	159E	156E-162E
28	015W	018W-012W	58	165E	162E-168E
29	009W	012W-006W	59	171E	168E-174E
30	003W	006W-000E	60	177E	174E-180W

8.3.3 GCTP Spheroid Codes

```
Clarke 1866 (default)         (0)
Clarke 1880                   (1)
Bessel                        (2)
International 1967            (3)
International 1909            (4)
WGS 72                        (5)
Everest                       (6)
WGS 66                        (7)
GRS 1980                      (8)
Airy                          (9)
Modified Airy                 (10)
Modified Everest              (11)
WGS 84                        (12)
Southeast Asia                (13)
Australian National           (14)
Krassovsky                         (15)
Hough                         (16)
Mercury 1960                  (17)
Modified Mercury 1968         (18)
Sphereof Radius 6370997m      (19)
```

```
Sphereof Radius 6371228m      (20)
Sphereof Radius 6371007.181m  (21)
```

8.3.4 Projection Parameters

Table 8-3. Projection Transformation Package Projection Parameters

Array Elements ➔								
Code & Projection Id	1	2	3	4	5	6	7	8
0 Geographic								
1 U T M	Lon/Z	Lat/Z						
3 Albers Conical Equal Area	Smajor	Sminor	STDPR1	STDPR2	CentMer	OriginLat	Fe	Fn
4 Lambert Conformal C	Smajor	Sminor	STDPR1	STDPR2	CentMer	OriginLat	FE	FN
5 Mercator	Smajor	Sminor			CentMer	TrueScale	FE	FN
6 Polar Stereographic	Smajor	Sminor			LongPol	TrueScale	FE	FN
7 Polyconic	Smajor	Sminor			CentMer	OriginLat	FE	FN
9 Transverse Mercator	Smajor	Sminor	Factor		CentMer	OriginLat	FE	FN
11 Lambert Azimuthal	Sphere				CentLon	CenterLat	FE	FN
20 Hotin Oblique Merc A	Smajor	Sminor	Factor			OriginLat	FE	FN
20 Hotin Oblique Merc B	Smajor	Sminor	Factor	AziAng	AzmthPt	OriginLat	FE	FN
22 Space Oblique Merc A	Smajor	Sminor		IncAng	AscLong		FE	FN
22 Space Oblique Merc B	Smajor	Sminor	Satnum	Path			FE	FN
24 Interrupted Goode	Sphere							
97 CEA utilized by EASE gr Notes)	Smajor	Sminor			CentMer	TrueScale	FE	FN
98 BCEA utilized by EASE (see Notes)	Smajor	Sminor			CentMer	TrueScale	FE	FN

Table 8-4. Projection Transformation Package Projection Parameters Elements

	Array Element				
Code & Projection Id	**9**	**10**	**11**	**12**	**13**
0 Geographic					
1 U T M					
3 Albers Conical Equal Area					
4 Lambert Conformal C					
5 Mercator					
6 Polar Stereographic					
7 Polyconic					
9 Transverse Mercator					
11 Lambert Azimuthal					
20 Hotin Oblique Merc A	Long1	Lat1	Long2	Lat2	zero
20 Hotin Oblique Merc B					one
22 Space Oblique Merc A	PSRev	SRat	PFlag	HDF-EOS	zero
22 Space Oblique Merc B				HDF-EOS	one
24 Interrupted Goode					
31 & 99 Integerized Sinusoidal	NZone		RFlag		
97 CEA utilized by EASE grid (Notes)					
98 BCEA utilized by EASE grid Notes)					

Where,

Lon/Z Longitude of any point in the UTM zone or zero. If zero, a zone code must be specified.

Lat/Z Latitude of any point in the UTM zone or zero. If zero, a zone code must be specified.

Smajor Semi-major axis of ellipsoid. If zero, Clarke 1866 in meters is assumed. It is recommended that explicit value, rather than zero, is used for Smajor.

Sminor Eccentricity squared of the ellipsoid if less than one, if zero, a spherical form is assumed, or if greater than one, the semi-minor axis of ellipsoid. It should be noted that a negative sphere code should be used in order to have user specified Smajor and Sminor be accepted by GCTP, otherwise default ellipsoid Smajor and Sminor will be used.

Sphere Radius of reference sphere. If zero, 6370997 meters is used. It is recommended that explicit value, rather than zero, is used for Sphere.

STDPR1 Latitude of the first standard parallel

STDPR2 Latitude of the second standard parallel

CentMer Longitude of the central meridian

OriginLat Latitude of the projection origin

FE	False easting in the same units as the semi-major axis
FN	False northing in the same units as the semi-major axis
TrueScale	Latitude of true scale
LongPol	Longitude down below pole of map
Factor	Scale factor at central meridian (Transverse Mercator) or center of projection (Hotine Oblique Mercator)
CentLon	Longitude of center of projection
CenterLat	Latitude of center of projection
Long1	Longitude of first point on center line (Hotine Oblique Mercator, format A)
Long2	Longitude of second point on center line (Hotine Oblique Mercator, frmt A)
Lat1	Latitude of first point on center line (Hotine Oblique Mercator, format A)
Lat2	Latitude of second point on center line (Hotine Oblique Mercator, format A)
AziAng	Azimuth angle east of north of center line (Hotine Oblique Mercator, frmt B)
AzmthPt	Longitude of point on central meridian where azimuth occurs (Hotine Oblique Mercator, format B)
IncAng	Inclination of orbit at ascending node, counter-clockwise from equator (SOM, format A)
AscLong	Longitude of ascending orbit at equator (SOM, format A)
PSRev	Period of satellite revolution in minutes (SOM, format A)
SRat	Satellite ratio to specify the start and end point of x,y values on earth surface (SOM, format A -- for Landsat use 0.5201613)
PFlag	End of path flag for Landsat: 0 = start of path, 1 = end of path (SOM, frmt A)
Satnum	Landsat Satellite Number (SOM, format B)
Path	Landsat Path Number (Use WRS-1 for Landsat 1, 2 and 3 and WRS-2 for Landsat 4 and 5.) (SOM, format B)
Nzone	Number of equally spaced latitudinal zones (rows); must be two or larger and even
Rflag	Right justify columns flag is used to indicate what to do in zones with an odd number of columns. If it has a value of 0 or 1, it indicates the extra column is on the right (zero) left (one) of the projection Y-axis. If the flag is set to 2 (two), the number of columns are calculated so there are always an even number of columns in each zone.

Notes:

- Array elements 14 and 15 are set to zero.
- All array elements with blank fields are set to zero.

All angles (latitudes, longitudes, azimuths, etc.) are entered in packed degrees/ minutes/ seconds (DDDMMMSSS.SS) format.

The following notes apply to the Space Oblique Mercator A projection:

- A portion of Landsat rows 1 and 2 may also be seen as parts of rows 246 or 247. To place these locations at rows 246 or 247, set the end of path flag (parameter 11) to 1--end of path. This flag defaults to zero.
- When Landsat-1,2,3 orbits are being used, use the following values for the specified parameters:
  - Parameter 4 099005031.2
  - Parameter 5 128.87 degrees - (360/251 * path number) in packed DMS format
  - Parameter 9 103.2669323
  - Parameter 10 0.5201613
- When Landsat-4,5 orbits are being used, use the following values for the specified parameters:
  - Parameter 4 098012000.0
  - Parameter 5 129.30 degrees - (360/233 * path number) in packed DMS format
  - Parameter 9 98.884119
  - Parameter 10 0.5201613

8.3.5 Additional projections

The following notes apply for **BCEA and CEA projections,** and **EASE grid**:

Behrmann Cylindrical Equal-Area (BCEA) projection was used for 25 km global EASE grid. For this projection the Earth radius is set to 6371228.0m and latitude of true scale is 30 degrees. For 25 km global EASE grid the following apply:

```
Grid Dimensions:
      Width 1383
      Height 586
Map Origin:
      Column (r0) 691.0
      Row (S0) 292.5
      Latitude 0.0
      Longitude 0.0
Grid Extent:
      Minimum Latitude 86.72S
      Maximum Latitude 86.72N
      Minimum Longitude 180.00W
      Maximum Longitude 180.00E
      Actual grid cell size 25.067525km
```

Grid coordinates (r,s) start in the upper left corner at cell (0.0), with r increasing to the right and s increasing downward.

Although the projection code and name (tag) kept the same, BCEA projection was generalized to accept Latitude of True Scales other than 30 degrees, Central Meridian other than zero, and ellipsoid earth model besides the spherical one with user supplied radius. This generalization along with the removal of hard coded grid parameters will allow users not only subsetting, but also creating other grids besides the 25 km global EASE grid and having freedom to use different appropriate projection parameters. With the current version one can create the above mentioned 25 km global EASE grid of previous versions using:

```
Grid Dimensions:
     Width 1383
     Height 586
Grid Extent:
     UpLeft Latitude 86.72
     LowRight Latitude -86.72
     UpLeft Longitude -180.00
     LowRight Longitude 180.00
Projection Parameters:
     1) 6371.2280/25.067525 = 254.16263
     2) 6371.2280/25.067525 = 254.16263
     5) 0.0
     6) 30000000.0
     7) 691.0
     8) -292.5
```

Also one may create **12.5 km global EASE grid** using:

```
Grid Dimensions:
     Width 2766
     Height 1171
Grid Extent:
     UpLeft Latitude 85.95
     LowRight Latitude -85.95
     UpLeft Longitude -179.93
     LowRight Longitude 180.07
Projection Parameters:
     1) 6371.2280/(25.067525/2) = 508.325253
     2) 6371.2280/(25.067525/2) = 508.325253
     5) 0.0
     6) 30000000.0
     7) 1382.0
     8) -585.0
```

Any other grids (normalized pixel or not) with generalized BCEA projection can be created using appropriate grid corners, dimension sizes, and projection parameters. Please note that like other projections Semi-major and Semi-minor axes will default to Clarke 1866 values (in meters) if they are set

A new projection CEA (97) was added to GCTP. This projection is the same as the generalized BCEA, except that the EASE grid produced will have its corners in meters rather than packed degrees, which is the case with EASE grid produced by BCEA.

9. Implementation of HDF-EOS 5

9.1 Software implementation

HDF-EOS 5 was first released to the public in 2001. It was developed as a contractual requirement under the NASA Earth Observing System Data and Information System Program by L-3 Communications. The software is currently supported under the EOS Maintenance and Development (EMD) Contract. The most current release of the software library was during December, 2005 in conjunction with HDF5-1.6.5. The software is supported on the following:

Operating Systems: Solaris (8, 9, 10), Irix6.5, HP 11, AIX, DEC, Windows NT/98/2000/XP, Linux (including 64-bit Opteron and Itanium), Mac OS X
Compilers: FORTRAN 77/90 & g77/pgf90 , C, C++, gcc, g++

Access to libraries and applications can be found at:

http://newsroom.gsfc.nasa.gov/sdptoolkit/toolkit.html
http://newsroom.gsfc.nasa.gov/sdptoolkit/HEG/HEGHome.html

Contact information on access and usage can be had from:

larry.klein@sesda2.com
Abe_Taaheri@raytheon.com
Landover_PGSTLKIT@raytheon.com

9.2 Applications

Over the past 10 years, dozens of applications have been written to access, browse, process and analyze data written in HDF-EOS2 and HDF-EOS 5 formats. Many of these applications have been converted to read and process HDF-EOS 5. Applications that have been provided and are supported by the EMD Program are:

- HDFView, a Java-based browser providing HDF4, HDF5, HDF-EOS 2 and 5 access.
- heconvert, which converts HDF-EOS 2 - based Grid/Point/Swath structures to HDF-EOS 5 equivalents).
- HE5View, a browser for viewing HDF-EOS5 files.
- HDF-EOS to GeoTIFF converter (HEG). This tool also provides subsetting, reprojection, stitching, etc.) GeoTIFF output is assessable to Geographical Information System (GIS) tools, ARCInfo, ERDAS and ENVI.

The commercial data analysis tools, IDL and Matlab also support HDF-EOS 5 files.

Other applications that provide specialized functionality to data products written in HDF-EOS 5 can be found at:
http://daac.gsfc.nasa.gov/
http://eosweb.larc.nasa.gov/
http://hdf.ncsa.uiuc.edu/hdfeoss.html

10. Operational Experience

The EOS Aura mission is designed to produce, archive and disseminate measurements of atmospheric constituents such as ozone, carbon monoxide and aerosols. As previously pointed out, The Aura team adopted HDF-EOS 5 as its' format of choice. Aura instrument measurements are stored in a common format developed before mission launch. Data from three of the four Aura instruments are archived at the Goddard Space Flight Center's GES Distributed Active Archive Center (DAAC). Detailed information on products and services can be found at: http://daac.gsfc.nasa.gov/.

The instrument represented at GSFC are: MLS, OMI and HRDLS. Currently the DAAC holds about 56,000 data granules and about 3.5 Terabytes of data volume. About 12 GBytes per day are ingested into the archives. So far total of 35 data products are available. The data are distributed electronically by user request.

Data from the TES instrument ate archived at the Langley Research Center DAAC. Detailed information can be found at: http://eosweb.larc.nasa.gov/. There are currently about 5000 data granules stored in about 2 Terabytes. 36 data products are available.

The MODIS and AIRS instrument teams from the EOS Terra and Aqua missions have produced data in the earlier HDF-EOS 2 format. Both teams have studied the costs and process of conversion of HDF-EOS 2 to HDF-EOS 5 format. At this time no decision has been made by either team on whether to proceed with this conversion during a future re-processing of data. This process could potentially put Petabytes of data into HDF-EOS 5 format.

Since introduction, more than 400 users have downloaded HDF-EOS and associated application software and on average 5 to 15 new users request passwords for downloading every month. These users will be supported by NASA for the indefinite future.

11. References

1. Release 7 SDP Toolkit Users Guide for the EMD Project for Toolkit Version 5.2.13, Document 333-EMD-001, Revision 03, April 2005, http://newsroom.gsfc.nasa.gov/sdptoolkit/userguide.html

2. HDF-EOS Interface Based on HDF5, Version 1.6.9, Volume 1: Overview and Examples, Document 175-EMD-001, Revision 03, April 2005, http://newsroom.gsfc.nasa.gov/sdptoolkit/userguide.html

3. HDF-EOS Interface Based on HDF5, Version 1.6.9, Volume 2: Function Reference Guide, Document 175-EMD-002, Revision 03, April 2005, http://newsroom.gsfc.nasa.gov/sdptoolkit/userguide.html

4. HDF5 User Documentation Release 1.6.5, November 2005 http://hdf.ncsa.uiuc.edu/HDF5/doc/

5. HDF5 for HDF4 Users: a short guide, National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, December 3, 2002, http://www.hdfgroup.uiuc.edu/papers/papers/h4toh5/HDF5forHDF4Users.pdf)

6. HDF5 Draft Community Standard, ESE RFC, 2005

APPENDIX A Structural Metadata Attributes

The following table describes briefly the attributes in the Structural Metadata for HDF-EOS5's Grid, Swath, and Point structures and shows the units if not self explanatory.

Table A-1. Summary of Structural Metadata Attributes

Structure	**Attribute Name**	**Definition**	**Fields Defining Attribute**	**datatype**
Swath				
	SwathName	Name of swath structure		String
	Dimension_X	Defined dimmension # X, with the name and size given by:		
			DimensionName	String
			Size	Integer
	DimensionMap_X	Defined dimension map # X, showing mapping between dimensions with the names specified in "GeoDimension" and "DataDimension". The positive "Offset" value shows how far along the data dimension one must travel to find the first point to have corresponding entry along the geolocation dimension. The negative "Offset" value shows how far along the geolocation dimension one must travel to find the first point to have corresponding entry along the data dimension. The "Increment" shows how many point to travel along the data dimension before the next point is found for which there is a corresponding entry along the geolocation dimension.		
			GeoDimension	String
			DataDimension	String
			Offset	Integer
			Increment	Integer

Table A-1. Summary of Structural Metadata Attributes(continued)

	IndexDimensionMap_X	Shows index mapping between fields given by GeoDimension and DataDimension. The indicies will be in a dataset with the name : _INDXMAP:geodim/datadim		
			GeoDimension	String
			DataDimension	String
	GeoField_X	Geo filed # X with the name, datatype and comma separated list of dimensions given by:		
			GeoFieldName	String
			DataType	Code
			DimList	String
			MaxDimList	String
		See Table 7.1 for compression codes.	CompressionType	Code
		Level of compression when the compression type is HE5_HDFE_COMP_DEFLATE. Compression parameter is in the range of 1 to 9.	DefelateLevel	Integer
		Pixels per block size and used for SZIP compression, an even number with typical values of 8, 10, 16, 32.	BlockSize	Integer
		Compression parameters for NBIT compression.	CompressionParams	Integer Array
	DataField_X	Data filed # X with the name, datatype and comma-separated list of dimensions, given by:		
			DataFieldName	String
		HDF5 datatype codes (See Appendix B of HDF5 Draft Community Standard, ESE RFC 007)	DataType	Code
			DimList	String
			MaxDimList	String
		See above for Geofields.	CompressionType	Code
		See above for Geofields.	DefelateLevel	Integer
		See above for Geofields.	BlockSize	Integer
		See above for Geofields.	CompressionParams	Integer Array

Table A-1. Summary of Structural Metadata Attributes(continued)

Grid				
	GridName	Name of grid structure. The grid is identified by:		String
		Number of columns	Xdim	Long Integer
		Number of rows	Ydim	Long Integer
		Upper left coordinates of the grid: The unit is meters for all projections except the "Geographic" projection. For the "Geographic" projection it is in packed degree/minutes/seconds (DDDMMMSSS.SS) format.	UpperLeftPointMtrs	Double
		Lower right coordinates of the grid: The unit is meters for all projections except the "Geographic" projection. For the "Geographic" projection it is in packed degree/minutes/seconds (DDDMMMSSS.SS) format.	LowerRightMtrs	Double
		The pojection code for the defined projection. The code is constructed as HE5_<name>, where name is the projection name given in section 8.3.1, such as HE5_GCTP_TM for the Transverse Mercator projection.	Projection	Code
		Projection parameters for the specified projection. See Table 8.3 for the elements in the array.	ProjParams	Double
		When the pixel registration is HE5_HDFE_CORNER the GridOrigin indicates which pixel corner represents the pixel location; The values can be HE5_HDFE_GD_UL, HE5_HDFE_GD_UR, HE5_HDFE_GD_LL, or HE5_HDFE_GD_LR. Any values in this attribute will be irrelevent if the pixel registration is defined as HE5_HDFE_CENTER.	GridOrigin	Code
		GCTP Sphere code (see 8.3.3)	SphereCode	Integer
		The pixel registration: HE5_HDFE_CENTER or HE5_HDFE_CORNER depending on the location in pixel that represents the pixel (see also GridOrigin above)	PixelRegistration	Code

Table A-1. Summary of Structural Metadata Attributes(continued)

		Zone code for the Universal Transverse Mercator (UTM) projection. See section 8.3.2.for the defined zone codes.	ZoneCode	Integer
	DataField_X	Data filed # X with the name, datatype and comma separated list of dimensions given by:		
			DataFieldName	String
		HDF5 datatype codes (See Appendix B of HDF5 Draft Community Standard, ESE RFC 007)	DataType	Code
			DimList	String
			MaxDimList	String
		See above for swath Geofields.	CompressionType	Code
		See above for swath Geofields.	DefelateLevel	Integer
		See above for swath Geofields.	BlockSize	Integer
		See above for swath Geofields.	CompressionParams	Integer Array
	Dimension_X	Defined dimmension # X, with the name and size:		
			DimensionName	String
			Size	Integer
Point				
	PointName	Name of the Point Structure.		String
	LevelName	Name of the level within the point structure.		String
	PointField_X	Point Field # X identified by PointFieldName, DataType, and Order of the field.		
			PointFieldName	String
			DataType	Code
		The dimension of the field. The order for numerical scaler variables can be either 0 or 1.	Order	Long Integer
	LevelLink_X	LevelLink # X identified by Parent and Child level names and the field name that links the levels together.		
			Parent	String
			Child	String
			LinkField	String
ZA		This structure is similar to Swath structure except that it does not have geofields and, therefore, any type of geofield-datafield mapping		
	ZaName	Name of ZA structure.		String

APPENDIX B Example HDF-EOS5 Swath Output

The following code fragment shows contents of HDF output created by the example code in Section 8.1.4. It shows how HDF-EOS5 swath objects are mapped onto HDF5 objects.

```
HDF5 "Swath.he5" {
GROUP "/" {
   GROUP "HDFEOS" {
      GROUP "ADDITIONAL" {
         GROUP "FILE_ATTRIBUTES" {
         }
      }
      GROUP "SWATHS" {
         GROUP "Swath1" {
            GROUP "Data Fields" {
               DATASET "Temperature" {
                  DATATYPE  H5T_IEEE_F64BE
                  DATASPACE  SIMPLE { ( 40, 20 ) / ( 40, 20 ) }
                  DATA {
                  (0,0): -999,-999,-999,-999,-999,-999,-999,-999,
                  (0,8): -999,-999,-999,-999,-999,-999,-999,-999,
                  (0,16): -999,-999,-999,-999,
                  (1,0): -999,-999,-999,-999,-999,-999,-999,-999,
                  (1,8): -999,-999,-999,-999,-999,-999,-999,-999,
                  (1,16): -999,-999,-999,-999,
                   ...........
                   ...........
                   ...........
                  (38,0): -999,-999,-999,-999,-999,-999,-999,-999,
                  (38,8): -999,-999,-999,-999,-999,-999,-999,-999,
                  (38,16): -999,-999,-999,-999,
                  (39,0): -999,-999,-999,-999,-999,-999,-999,-999,
                  (39,8): -999,-999,-999,-999,-999,-999,-999,-999,
                  (39,16): -999,-999,-999,-999
                  }
                  ATTRIBUTE "_FillValue" {
                     DATATYPE  H5T_IEEE_F64BE
                     DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
                     DATA {
                     (0): -999
                     }
                  }
                  ATTRIBUTE "Unit" {
                     DATATYPE  H5T_STRING {
                           STRSIZE 13;
                           STRPAD H5T_STR_NULLTERM;
                           CSET H5T_CSET_ASCII;
                           CTYPE H5T_C_S1;
                        }
                     DATASPACE  SCALAR
                     DATA {
                     (0): "Degree Kelvin"
                     }
                  }
               }
            }
```

```
            GROUP "Geolocation Fields" {
               DATASET "Time" {
                  DATATYPE  H5T_IEEE_F64BE
                  DATASPACE  SIMPLE { ( 20 ) / ( 20 ) }
                  DATA {
                  (0): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
                  (18): 0, 0
                  }
                  ATTRIBUTE "_FillValue" {
                     DATATYPE  H5T_IEEE_F64BE
                     DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
                     DATA {
                     (0): 0
                     }
                  }
               }
            }
         }
         GROUP "Swath2" {
            GROUP "Data Fields" {
            }
            GROUP "Geolocation Fields" {
            }
            DATASET "_INDEXMAP:IndexTrack,Res2tr_indexed" {
               DATATYPE  H5T_STD_I32BE
               DATASPACE  SIMPLE { ( 6 ) / ( 6 ) }
               DATA {
               (0): 1, 5, 8, 12, 17, 20
               }
            }
         }
      }
   }
   GROUP "HDFEOS INFORMATION" {
      ATTRIBUTE "HDFEOSVersion" {
         DATATYPE  H5T_STRING {
               STRSIZE 32;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "HDFEOS_5.1.10"
         }
      }
      DATASET "StructMetadata.0" {
         DATATYPE  H5T_STRING {
               STRSIZE 32000;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "GROUP=SwathStructure
```

```
GROUP=SWATH_1
      SwathName="Swath1"
      GROUP=Dimension
            OBJECT=Dimension_1
                  DimensionName="GeoTrack"
                  Size=20
            END_OBJECT=Dimension_1
            OBJECT=Dimension_2
                  DimensionName="GeoXTrack"
                  Size=10
            END_OBJECT=Dimension_2
            OBJECT=Dimension_3
                  DimensionName="Res2tr"
                  Size=40
            END_OBJECT=Dimension_3
            OBJECT=Dimension_4
                  DimensionName="Res2xtr"
                  Size=20
            END_OBJECT=Dimension_4
            OBJECT=Dimension_5
                  DimensionName="Bands"
                  Size=15
            END_OBJECT=Dimension_5
            OBJECT=Dimension_6
                  DimensionName="ProfDim"
                  Size=4
            END_OBJECT=Dimension_6
            OBJECT=Dimension_7
                  DimensionName="Unlim"
                  Size=-1
            END_OBJECT=Dimension_7
      END_GROUP=Dimension
      GROUP=DimensionMap
            OBJECT=DimensionMap_1
                  GeoDimension="GeoTrack"
                  DataDimension="Res2tr"
                  Offset=0
                  Increment=2
            END_OBJECT=DimensionMap_1
            OBJECT=DimensionMap_2
                  GeoDimension="GeoXTrack"
                  DataDimension="Res2xtr"
                  Offset=1
                  Increment=2
            END_OBJECT=DimensionMap_2
      END_GROUP=DimensionMap
      GROUP=IndexDimensionMap
      END_GROUP=IndexDimensionMap
      GROUP=GeoField
            OBJECT=GeoField_1
                  GeoFieldName="Time"
                  DataType=H5T_NATIVE_DOUBLE
                  DimList=("GeoTrack")
                  MaxdimList=("GeoTrack")
            END_OBJECT=GeoField_1
      END_GROUP=GeoField
      GROUP=DataField
```

```
                        OBJECT=DataField_1
                              DataFieldName="Temperature"
                              DataType=H5T_NATIVE_DOUBLE
                              DimList=("Res2tr","Res2xtr")
                              MaxdimList=("Res2tr","Res2xtr")
                        END_OBJECT=DataField_1
                  END_GROUP=DataField
                  GROUP=ProfileField
                  END_GROUP=ProfileField
                  GROUP=MergedFields
                  END_GROUP=MergedFields
            END_GROUP=SWATH_1
            GROUP=SWATH_2
                  SwathName="Swath2"
                  GROUP=Dimension
                        OBJECT=Dimension_1
                              DimensionName="Res2tr_indexed"
                              Size=40
                        END_OBJECT=Dimension_1
                        OBJECT=Dimension_2
                              DimensionName="IndexTrack"
                              Size=6
                        END_OBJECT=Dimension_2
                  END_GROUP=Dimension
                  GROUP=DimensionMap
                  END_GROUP=DimensionMap
                  GROUP=IndexDimensionMap
                        OBJECT=IndexDimensionMap_1
                              GeoDimension="IndexTrack"
                              DataDimension="Res2tr_indexed"
                        END_OBJECT=IndexDimensionMap_1
                  END_GROUP=IndexDimensionMap
                  GROUP=GeoField
                  END_GROUP=GeoField
                  GROUP=DataField
                  END_GROUP=DataField
                  GROUP=ProfileField
                  END_GROUP=ProfileField
                  GROUP=MergedFields
                  END_GROUP=MergedFields
            END_GROUP=SWATH_2
           END_GROUP=SwathStructure
           GROUP=GridStructure
           END_GROUP=GridStructure
           GROUP=PointStructure
           END_GROUP=PointStructure
           GROUP=ZaStructure
           END_GROUP=ZaStructure
           END
           "
         }
      }
   }
}
}
```

APPENDIX C Example HDF-EOS5 Grid Output

The following code fragment shows contents of HDF output created by the example code in Section 8.2.4. It shows how HDF-EOS5 grid objects are mapped onto HDF5 objects.

```
HDF5 "Grid.he5" {
GROUP "/" {
   GROUP "HDFEOS" {
      GROUP "ADDITIONAL" {
         GROUP "FILE_ATTRIBUTES" {
         }
      }
      GROUP "GRIDS" {
         GROUP "TMGrid" {
            GROUP "Data Fields" {
               DATASET "Voltage" {
                  DATATYPE  H5T_IEEE_F32BE
                  DATASPACE  SIMPLE { ( 5, 7 ) / ( 5, 7 ) }
                  DATA {
                  (0,0): -1.11111,-1.11111,-1.11111,-1.11111,-1.11111,
                  (0,5): -1.11111,-1.11111,
                  (1,0): -1.11111,-1.11111,-1.11111,-1.11111,-1.11111,
                  (1,5): -1.11111,-1.11111,
                  (2,0): -1.11111,-1.11111,-1.11111,-1.11111,-1.11111,
                  (2,5): -1.11111,-1.11111,
                  (3,0): -1.11111,-1.11111,-1.11111,-1.11111,-1.11111,
                  (3,5): -1.11111,-1.11111,
                  (4,0): -1.11111,-1.11111,-1.11111,-1.11111,-1.11111,
                  (4,5): -1.11111,-1.11111
                  }
                  ATTRIBUTE "_FillValue" {
                     DATATYPE  H5T_IEEE_F32BE
                     DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
                     DATA {
                     (0): -1.11111
                     }
                  }
               }
            }
         }
      }
   }
   GROUP "HDFEOS INFORMATION" {
      ATTRIBUTE "HDFEOSVersion" {
         DATATYPE  H5T_STRING {
               STRSIZE 32;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "HDFEOS_5.1.10"
         }
      }
      DATASET "StructMetadata.0" {
```

```
         DATATYPE  H5T_STRING {
               STRSIZE 32000;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "GROUP=SwathStructure
           END_GROUP=SwathStructure
           GROUP=GridStructure
            GROUP=GRID_1
                  GridName="TMGrid"
                  XDim=5
                  YDim=7
                  UpperLeftPointMtrs=(4855670.775390,9458558.924830)
                  LowerRightMtrs=(5201746.439830,-10466077.249420)
                  Projection=HE5_GCTP_TM
                  ProjParams=(0,0,0.999600,0,-75000000,0,5000000,
                  0,0,0,0,0,0)
                  SphereCode=0
                  GROUP=Dimension
                        OBJECT=Dimension_1
                              DimensionName="Time"
                              Size=10
                        END_OBJECT=Dimension_1
                        OBJECT=Dimension_2
                              DimensionName="Unlim"
                              Size=-1
                        END_OBJECT=Dimension_2
                  END_GROUP=Dimension
                  GROUP=DataField
                        OBJECT=DataField_1
                              DataFieldName="Voltage"
                              DataType=H5T_NATIVE_FLOAT
                              DimList=("XDim","YDim")
                              MaxdimList=("XDim","YDim")
                        END_OBJECT=DataField_1
                  END_GROUP=DataField
                  GROUP=MergedFields
                  END_GROUP=MergedFields
            END_GROUP=GRID_1
           END_GROUP=GridStructure
           GROUP=PointStructure
           END_GROUP=PointStructure
           GROUP=ZaStructure
           END_GROUP=ZaStructure
           END
           "
         }
      }
   }
}
}
```